Oracle® Communications Billing and Revenue Management

Configuring Pipeline Rating and Discounting

Release 12.0
E51019-13
September 2024

ORACLE

Oracle Communications Billing and Revenue Management Configuring Pipeline Rating and Discounting, Release 12.0

E51019-13

Contents

3 Configuring EDR Input Processing

4 Configuring EDR Output Processing

5 Configuring EDR Preprocessing

6 Setting Up EDR Enrichment

7 Setting Up Pipeline Aggregation

8 Migrating Pipeline Manager Data between Test and Production Systems

9 Transferring Data Between Pipeline Manager Databases

10 Creating iScripts and iRules

Part II Setting Up Pricing for Real-Time Rating and Pipeline Manager

11 About Creating a Price List

12 Understanding the Sample Pricing Plans

13 Setting Up Real-Time and Pipeline Pricing and Rating

14 Configuring Resource Rounding

15 Managing Sub-Balances

16 About Real-Time Rating

17 About Pipeline Rating

18 Policy-Driven Charging

19 Working with Promotions

20 Working with Provisioning Tags

21 Working with Extended Rating Attributes

22 Rating Implementation and Customization

23 Testing Your Price List

24 Using the XML Pricing Interface to Create a Price List

25 XML Examples of Creating Pricing Components

30 Real-Time Rating Based on Event or Resource Quantity

31 Setting Up Zones for Batch Pipeline Rating

32 Real-Time Rating Using Event Attributes

33 Migrating Pricing Data from Legacy Databases

41 Setting Up Pipeline Manager Taxation

42 Credit Limit and Threshold Checking during Batch Rating

47 About Suspense Manager

48 Installing and Configuring Suspense Manager

52 Suspense Management Utilities

53 Recycling EDRs in Pipeline-Only Systems

Part VI Loading Rated Events

54 Understanding Rated Event Loader

55 Installing Rated Event Loader

56 Configuring Rated Event Loader

57 Loading Prerated Events

59 About Real-Time Rerating Pipelines

60 About Rerating Pipeline-Rated Events

61 Using Event Extraction Manager

62 Configuring Rerating in Pipeline Manager

63 About Comprehensive Rerating Using pin_rerate

64 Configuring Comprehensive Rerating

65 Using the pin_rerate Utility

68 Configuring Pipeline Manager

69 Starting and Stopping Pipeline Manager

70 Monitoring Pipeline Manager

71 Optimizing Pipeline Manager Performance

72 Migrating Accounts with the Pipeline Manager Running

73 Pipeline Manager Error Messages

75 Testing Pipeline Manager

Part X Installing Account Synchronization

76 About Sending Account Data to Pipeline Manager

77 Installing and Configuring Account Synchronization

82 Pipeline Manager Data Modules

83 Pipeline Manager iRules

84 Pipeline Manager iScripts

85 Pipeline Manager iScript Functions

86 Pipeline Manager Input and Output Modules

87 Pipeline Manager Framework Modules

88 Pipeline Manager Utilities

89 Pipeline Manager Configuration File Reference

90 Pipeline Manager Opcode Reference

91 Event Notification Definitions

92 Revenue Assurance Manager Reports

Preface

The book describes Oracle Communications Billing and Revenue Management pipeline rating.

Audience

This book is intended for system administrators and developers.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Part I
Configuring Pipeline Rating

This part describes how to configure pipeline rating in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

1

About Pipeline Rating

This chapter describes how to configure pricing components for rating with Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

About Configuring Pipeline Rating

Configuring pipeline rating involves two sets of tasks:

- Creating pricing components by using Pricing Center or PDC.
- Configuring rating function modules.

Rating EDRs Split across Time Periods

Event data records (EDRs) sometimes overlap time periods. For example, if off-peak rating starts at 7:30 p.m., and a call begins at 7:10 p.m. and ends at 7:35 p.m., the call overlaps the boundary between peak and off-peak rates. When you define a charge, you specify a Splitting option that determines how an EDR that overlaps a time period is rated.

You can choose from four splitting options:

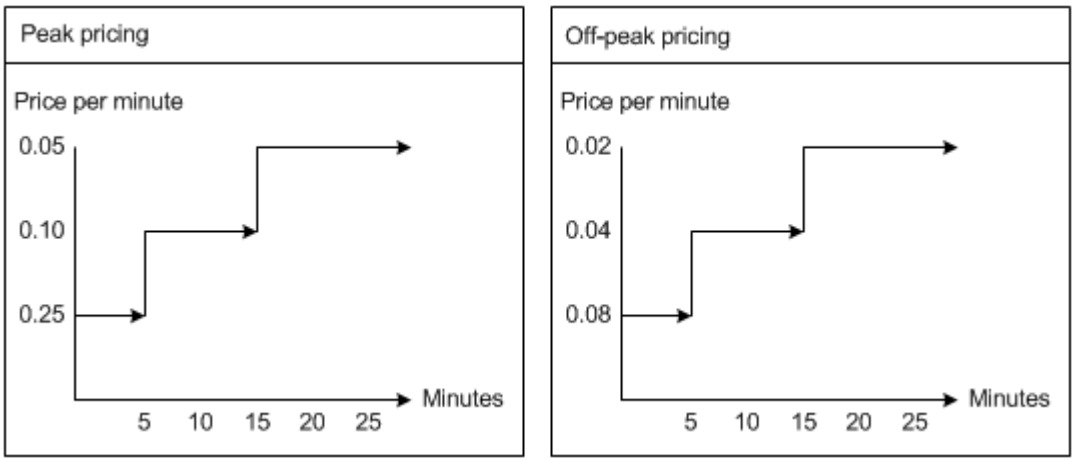
- No splitting, based on start time: Use the start time to rate the entire call.
- No splitting, based on end time: Use the end time to rate the entire call.
- Consecutive splitting: Split the call into separately rated parts. When using multiple price steps for different usage levels, continue counting the call duration from the time of the split.
- Isolated splitting: Split the call into separately rated parts. When using multiple price steps for different usage levels, start over at zero from the time of the split.

The Consecutive and Isolated splitting options are used when you use a combination of price steps and time-based rating. With this combination, you can have a case where an event is split by time zones, resulting in the application of two different pricings, each with its own Pricing Steps. The Consecutive and Isolated splitting options enable you to determine how to handle the selection of the appropriate Pricing Step.

Here is an example:

Figure 1-1 shows the peak pricing and the off-peak pricing:

Figure 1-1 Peak and Off-Peak Pricing



06:00 - 07:30 is peak

07:30 - 10:00 is off-peak

An EDR is rated that starts at 7:10 and finishes 7:35, for a total of 25 minutes. The event is split into two segments:

07:10 – 07:30 - peak (20 minutes)

07:30 – 07:35 - off-peak (5 minutes)

Figure 1-2 shows how Consecutive splitting works. The gray areas show how the pricing steps are applied in both peak and off-peak pricing. In this case, the first 20 minutes apply to the peak pricing, so the call is rated at .25 per minute, .10 per minute, and .5 per minute. The last 5 minutes are rated by the off-peak pricing, which, since it uses consecutive splitting, takes into account the previous 20 minutes, and rates the final 5 minutes at .02 per minute:

Figure 1-2 Consecutive Splitting

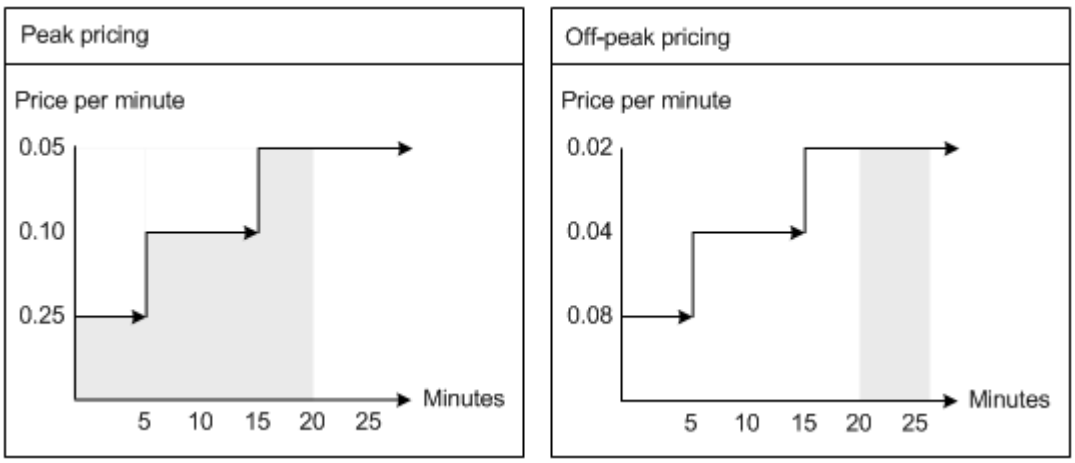
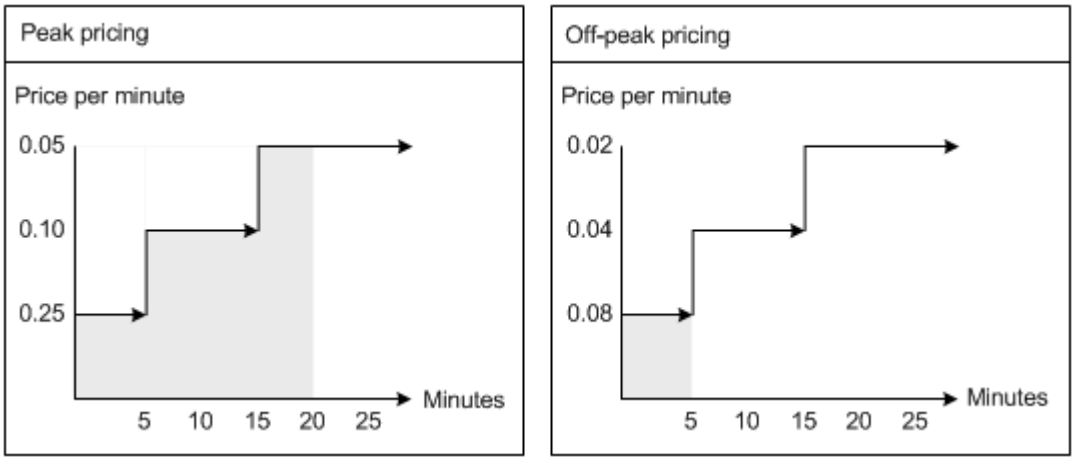


Figure 1-3 shows how Isolated splitting works. In this case, the first 20 minutes apply to the peak Pricing in the same way as Consecutive splitting. The last 5 minutes are rated by the off-peak model, which, because it uses Isolated splitting, does not consider the previous 20

minutes. Instead, it begins counting at zero, so the remaining 5 minutes are rated by the first step, that is, .08 per minute:

Figure 1-3 Isolated Splitting



Using Consecutive splitting and Isolated splitting gives different results for the final charge for the EDR. In this example:

Table 1-1 shows the results of **Consecutive splitting**:

Table 1-1 Results of Consecutive Splitting

Usage	Calculation	Result
Peak (20 minutes)	(5 * 0.25) + (15 * 0.10) + (5 * 0.05)	3.00
Off-peak (5 minutes)	5 * 0.02	0.10
NA	Total	3.10

Table 1-2 shows the results of **Isolated splitting**:

Table 1-2 Results of Isolated Splitting

Usage	Calculation	Result
Peak (20 minutes)	(5 * 0.25) + (15 * 0.10) + (5 * 0.05)	3.00
Off-peak (5 minutes)	5 * 0.08	0.40
NA	Total	3.40

About Pipeline Charge Versions

Each charge must have at least one charge version and can have as many versions as you need. Each charge version is valid for a different time period and only one version is valid for each period. The start time of an EDR determines which charge version is used for that record.

In addition to the validity period, you specify a zone model when you define a charge version. The zone model determines how calls to and from different regions are classified for rating. Each charge version can use different zone models.

Charge versions are stored in the IFW_RATEPLAN_VER table in the Pipeline Manager database.

Charge versions can be either basic or delta versions.

- A *basic version* can be used as the basis of other versions. You must specify all required attributes when you create a basic charge version.
- A *delta version* inherits attributes from a basic charge version that you specify when you create the delta version. In the delta version, you enter only the attributes that you want to change. For example, you might change only the **Version**, **Valid From,** and **Zone Model** fields.

About Pipeline Charge Configurations

The charge configuration determines which pricing or price selector is used to charge a given EDR. When you define a charge configuration, you specify a combination of criteria that an EDR must match to be rated by a particular pricing.

Each configuration maps a combination of service code, service class, and impact category to a combination of time model and time period. The configuration then maps the time model/time period combination to a pricing or price selector.

You can create any number of configurations for a charge version. The configurations in a given charge version must cumulatively cover all possible combinations of service code, service class, impact category, time model, and time period.

When you associate a pricing with a charge configuration, you can optionally specify an alternative pricing that is used in addition to the main pricing. You can compare the charges that result from the two models. For example, you may want to better understand the financial impact of a change to a different pricing before committing to the change.

Charge configurations are stored in the IFW_RATEPLAN_CNF table in the Pipeline Manager database.

Using Passthrough Prices

When you define a charge configuration, you can choose to ignore the calculated price and use a price that is passed in by the CDR instead. For example, you can ignore the calculated charge and use a passed-in charge if you receive external wholesale charges and want to use them for retail rating.

You can also modify or replace the passed-in price by specifying an add-on type and entering a charge. There are three add-on types:

- **Percentage** increases the passed-in price by a percentage that you enter.
- **Addon Value** increases the passed-in price by a fixed amount that you enter.
- **New Value** replaces the passed-in price with an amount you enter.

To use the passed-in price without modification, specify **0** as the charge.

About Pipeline Rate Adjustments

Rate adjustments are an optional way to customize a pipeline charge version. You can use rate adjustments to provide discounts based on date, time, service, and other event attributes. For example, you can provide a discount on all calls for a specific day.

An adjustment can be a percentage change to the original charge, a value to be added to the charge, or a completely new value to replace the charge.

When you define a rate adjustment, you specify dates and times during which the adjustment is valid and a maximum quantity above which the adjustment does not apply.

You also specify rules that determine whether an EDR qualifies for the adjustment. These rules filter EDRs based on values such as usage class, usage type, service code, service class, impact category, source network, and destination network. You can enter fixed values, expressions, or a wildcard (**.***) that matches all values.

When you use a rate adjustment, the original charge is overwritten by the adjusted charge. This is different from discounting, which leaves the original charge in place while calculating a discounted charge. Adjustments and discounts are also handled differently for accounting purposes. When calculating the general ledger (G/L) impact, the adjusted amount is not considered revenue. When you use discounting, however, the discounted amount is counted as revenue.

There are two ways to define rate adjustments:

- You can define rate adjustments in Pricing Center. In this case, the rate adjustment data is stored in the IFW_RATEADJUST table in the Pipeline Manager database.
- You can create a file that defines rate adjustment rules.

Rate adjustment is performed by the FCT_RateAdjust module. The module reads data from the EDR and evaluates it according to the rate adjustment rules stored in the database or in the rate adjustment file.

2

Configuring Pipeline Rating

This chapter describes how to configure the Oracle Communications Billing and Revenue Management (BRM) pricing data and modules necessary for batch rating with Pipeline Manager.

About Configuring Function Modules for Pipeline Rating

Three groups of modules are involved in rating:

- The pre-rating modules (FCT_ GlobalRating, FCT_CustomerRating, and FCT_RSC_Map) gather information and prepare the event data record (EDR) for rating.
- The FCT_MainRating module applies charges and pricing to the EDR, creating charge breakdown data, including charge packets with charges.
- The post-rating modules (FCT_RateAdjust and FCT_BillingRecord) make changes to the EDR after the charge data is included.

Multiple Pipeline Manager modules add and delete charge packet blocks to EDRs. Therefore, the value in the NUMBER_OF_CHARGE_PACKETS field does not reflect the actual number of charge packets in the charge breakdown record. You can get the correct number of charge packets by using the edrNumDatablocks function in a custom iScript module.

Depending on how your pipelines are set up, you need to configure some or all of these modules for rating.

About the Rating Data Modules

Configure the following data modules for rating:

- DAT_AccountBatch
- DAT_BalanceBatch
- DAT_Calendar
- DAT_Currency
- DAT_Dayrate
- DAT_ExchangeRate
- DAT_ItemAssign
- DAT_ModelSelector
- DAT_NOSP
- DAT_PriceModel
- DAT_Rateplan
- DAT_TimeModel

About Using Filter Sets to Apply System Products and Discounts

Use filter sets to apply system products/charge offers and system discounts/discount offers to a select group of customers. For example, you can provide reduced international call rates for all customers with great credit.

When configured to use filter sets, the pipeline:

1. Analyzes each EDR to determine whether it meets the criteria for a filter set.
2. Adds any applicable system products/charge offers and system discounts/ discount offers, along with their priorities, to the EDR's list of purchased products/charge offers.
3. Uses the product/charge offer and/or discount/discount offer when rating the EDR.

> **Note:**
>
> The actual product/charge offer or discount/discount offer the pipeline uses to rate the EDR depends on how your system is configured.

You use the following to configure your system for filter sets:

- **system_filterset_edr_field_values**. This file specifies which EDR fields and values can be used as filter criteria. You must load this file into the BRM database prior to creating your filter sets.
- **IRL_UsageType**. This iRule assigns usage types to an EDR. This signals the pipeline that the EDR qualifies for the special consideration that a filter set contains.
- **FCT_Filter_Set**. This module determines whether an EDR qualifies for any filter sets, and, if it does, adds any applicable system products/charge offers and system discounts/ discount offers to the EDR's list of purchased products/charge offers.

To configure BRM to use filters sets, perform the following tasks:

1. Specify which EDR fields and values can be used as filter criteria and then load the data into the BRM database. See "Loading Filter Set Data into BRM".
2. Define your filter sets by using a custom application. See "Defining Your Filter Sets".
3. Configure the IRL_UsageType iRule to assign usage types to EDRs.
4. Configure FCT_Filter_Set to apply system products/charge offers and discount/discount offers to specified market segments. See "FCT_Filter_Set".
5. Connect the FCT_DiscountAnalysis module to the FCT_Filter_Set module. Use the FCT_DiscountAnalysis module's **Filter_SetModule** registry entry. See "FCT_DiscountAnalysis".

Loading Filter Set Data into BRM

You can specify the EDR fields and values that you can use as filter criteria. To do so, you edit the **system_filterset_edr_field_values.en_US** sample file in the *BRM_home***/sys/msgs/system_filter_set** directory. *BRM_home* is the directory where you installed BRM components. For example, to use location as filter criteria, add these entries to the file:

```
DETAIL.ASS_CAMEL_EXT.MSC_ADDRESS="London"
DETAIL.ASS_CAMEL_EXT.MSC_ADDRESS="Paris"
```

After defining the field values, you use the load_localized_strings utility to load the contents of the **system_filterset_edr_field_values.***locale* file into the **/strings** object. To run the **load_localized_strings** utility, use this command:

```
load_localized_strings system_filterset_edr_field_values.locale
```

> **Note:**
>
> If you are loading a localized version of this file, use the correct file extension for your locale.

Defining Your Filter Sets

You define your filter sets by creating a custom client application to call the filter set opcodes. BRM then stores data about each filter set in **/filter_set/product** objects in the BRM database.

> **Note:**
>
> To create and manage filter sets by using a custom client application, configure your application to call the filter set opcodes.

To create a filter set, perform the following:

1. Create your system products/charge offers and system discounts/discount offers and associate them with specific service and event types. For example:
   - Create a system charge offer for **/service/telco/gsm/telephony** events that charges a flat rate of 5 cents per minute.
   - Create a system discount offer for **/service/telco/gsm/sms** events that provides a 20% discount for the first 10 minutes of usage.
2. Create your filter sets by mapping filter criteria to system charge offers and system discount offers. When creating filter sets, you specify:
   - The conditions required for an EDR to qualify for a system charge offer or discount offer. That is, the list of required EDR fields and values for each filter set.
   - Applicable system products/charge offers and discounts/discount offers, along with their priority and validity dates.

   In Table 2-1, Filter_1 specifies that all GSM wireless calls made to Paris by customers with great credit qualify for a 20% discount.

Table 2-1 Example Filters

Filter name	Filter criteria	System charge offer or system discount offer	Priority
Filter_1	DETAIL.ASS_GSMW_EXT.RECORD_TYPE="Great credit" DETAIL.ASS_CAMEL_EXT.MSC_ADDRESS="Paris"	20% Discount	3

Table 2-1 (Cont.) Example Filters

Filter name	Filter criteria	System charge offer or system discount offer	Priority
Filter_2	DETAIL.ASS_GSMW_EXT.RECORD_TYPE="Great credit"	Flat rate charge offer	6

About Global Rating

Use global rating to rate every EDR by using the same set of charges. Global rating is performed by the FCT_ GlobalRating module. This module adds an associated charge breakdown record to the EDR. Each charge, or partial charge, from the global charges adds a charge packet.

Global rating is typically used for gathering data used for business planning. For example:

- You can use global rating to calculate an average wholesale charge. You can then compare the average charge with the actual charge to find profit margins for different customer types.
- You can use global rating to calculate an average retail charge. You can use this data to monitor unusual charge amounts that can indicate an error in a rating configuration.

To configure global rating, see "FCT_CustomerRating". Use the **EdrRatePlans** registry entry to specify the global charges. You can use this entry in a semaphore.

About Least Cost Rating

Use least cost rating to rate EDRs by using the charge offer that produces the lowest charge to customers. In this configuration, the pipeline:

1. Rates EDRs by using all products/charge offers associated with an EDR and applies any discounts.
2. Finds the charge and discount that produces the lowest charge.
3. Applies the lowest charge to the customer's balance impact.

> **Note:**
>
> An EDR can qualify for *either* least cost rating or overlay promotion. It cannot qualify for both. See "About Overlay Promotions".

You use the following modules to find the lowest possible charge for an event:

- The IRL_LeastCostPerEDR iRule screens EDRs for least cost rating. It compares each EDR against conditions that you specify. When an EDR meets the criteria, the module flags the EDR for least cost rating.
- The FCT_CustomerRating module checks whether an EDR is flagged for least cost rating, and, if it is, generates breakdown records and associated charge packets for each charge offer. All charge packets are listed in order of priority, highest priority first.
- The ISC_LeastCost iScript calculates the total charge for each charge offer and discount and flags the charge that produces the lowest charge. When the lowest charge is

generated by a promotional charge offer, the module also calculates the total savings between the promotional charge offer and the lowest priority (base) charge offer.

You configure the modules in a discounting pipeline that includes the FCT_DiscountAnalysis module and the FCT_Discount module. These modules find any applicable discounts for each charge and calculate the discount for each charge packet.

Configuring Least Cost Rating

To set up least cost rating:

1. Specify your least cost rating criteria. See "Specifying the Rules to Qualify for Least Cost Rating". To do so, you edit and configure the IRL_LeastCostPerEDR iRule. See "IRL_LeastCostPerEDR".
2. Configure FCT_CustomerRating for least cost rating by using the **LeastCostRating** entry. See "FCT_CustomerRating".
3. Configure the ISC_LeastCost iScript to find the lowest charge for customers. See "ISC_LeastCost".

Specifying the Rules to Qualify for Least Cost Rating

To specify which EDRs are flagged for least cost rating:

1. Edit the *pipeline_home*/**iScriptLib/iScriptLib_Standard/IRL_LeastCostPerEDR.irl** file to include your rules for least cost rating. *pipeline_home* is the directory where you installed Pipeline Manager. The variables in the file correspond to positions in the **IRL_LeastCostPerEDR.data** file.

   For example, this rule specifies that EDRs matching the conditions in position 1 will have their DETAIL.CUST_A.LEAST_COST EDR field set to the value in position 2 of the **IRL_LeastCostPerEDR.data** file:

   ```
   CONDITION:

   SetLeastCostDefault ();
   ${1};

   RESULT:
   edrLong(DETAIL.CUST_A.LEAST_COST) = ${2};
   ```

2. Edit the *pipeline_home*/**iScriptLib/iScriptLib_Standard/IRL_LeastCostPerEDR.data** file to include the conditions required for least cost rating. Use Boolean operators to specify the combinations of market segments and charge offer priorities for testing whether EDRs qualify for least cost rating. You can create any number of entries.

   For example, this rule specifies that only EDRs that meet both of these criteria are flagged for least cost rating (DETAIL.CUST_A.LEAST_COST set to 2):

   - MARKET_SEGMENT EDR field = 1234
   - Charge offer priority greater than 4

   ```
   segmentContains("1234") and priority()>4;2
   ```

3. Configure the FCT_IRules module to point to **IRL_LeastCostPerEDR.irl** by using the following registry entries:

> **Note:**
>
> If you configure a pipeline to use filter sets, make sure IRL_LeastCostPerEDR runs *after* FCT_Filter_Set.

See "IRL_LeastCostPerEDR".

About Calculating the Promotional Savings

You can calculate how much money your customers save when an event is rated with a promotional charge offer rather than a base charge offer by using promotional savings. *Promotional savings* allows you to calculate the difference in charges between a promotional charge offer and the base charge offer, so you can advertise the savings to your customers.

When configured for promotional savings, the pipeline:

1. Rates EDRs by using the highest priority charge offer and the lowest priority (base) charge offer.
2. Calculates the difference between the charge for the promotional charge offer and the charge for the base charge offer.
3. Applies the savings amount to the EDR.

> **Note:**
>
> The pipeline applies the savings amount only when the promotional charge offer generates the lowest charge.

You use the following modules to calculate promotional savings:

- The IRL_PromotionalSavingPerEDR iRule screens EDRs for promotional savings. It compares each EDR against your specified criteria. When an EDR meets the criteria, the module flags the EDR for promotional savings.
- The FCT_CustomerRating module checks whether an EDR is flagged for promotional savings, and, if it is, generates separate charge breakdown records and associated charge packets for the highest priority (promotional) and lowest priority (base) charge offer.
- The ISC_LeastCost iScript calculates the charges for the highest and lowest priority products/charge offers and calculates the total savings amount.

See "How Pipeline Modules Process Overlay Promotions" for information about how the processing of overlay promotions and promotional savings are related.

To set up promotional savings:

1. Specify your promotional savings criteria. See "Specifying the Rules to Qualify for Promotional Savings".
2. Configure FCT_CustomerRating for promotional savings by using the **PromotionalSaving** entry. See "FCT_CustomerRating".
3. Configure ISC_LeastCost to calculate the promotional savings amount. See "ISC_LeastCost".

Specifying the Rules to Qualify for Promotional Savings

To specify which EDRs are flagged for a promotional savings calculation:

1. Edit the *pipeline_home*/**iScriptLib/iScriptLib_Standard/IRL_PromotionalSavingPerEDR.irl** file to include your rules for qualifying for a promotional savings calculation. The variables in the file correspond to positions in the **IRL_PromotionalSavingPerEDR.data** file.

   For example, this rule specifies that EDRs matching the conditions in position 1 will have their DETAIL.CUST_A.PROMOTIONAL_SAVING EDR field set to the value in position 2:

   ```
   CONDITION:

   ${1};

   RESULT:
   edrLong( DETAIL.CUST_A.PROMOTIONAL_SAVING ) = ${2};
   ```

2. Edit the *pipeline_home*/**iScriptLib/iScriptLib_Standard/IRL_PromotionalSavingPerEDR.data** file to include your conditions for qualifying for a promotional savings calculation. Use Boolean operators to specify the required combination of filter sets and charge offer priorities. You can create any number of entries.

   For example:

   ```
   productName("Standard GSM Telephony", "Highest") and priority("Highest") >
   4 and usageStartTimeGreaterThan("20040101000000", "Highest") and
   serviceType("/service/telco/gsm/telephony", "Highest"); 2

   productName("Standard GSM Telephony", "Base") and priority("Base") < 4 and
   usageStartTimeGreaterThan("20040101000000", "Base") and serviceType("/
   service/telco/gsm/telephony", "Base"); 2
   ```

3. Configure the FCT_IRules module to run the IRL_PromotionalSavingPerEDR iRule. Use the PromotionalSaving entry to specify the path to your IRL_PromotionalSavingPerEDR iRules file.

   For more information, see "IRL_PromotionalSavingPerEDR".

About Overlay Promotions

Overlay promotions allow you to quickly and easily replace your existing products/charge offers and discounts/discount offers with special products/charge offers and discounts/discount offers that take precedence. For example, you may want to offer a special 10% discount for three months to your existing customers who have paid all their bills on time. To do this, you add a charge offer that offers the same services but costs10% less and has a higher priority than the original charge offer.

You do not need to create new zone models for promotional products/charge offers. Promotional products/charge offers can use the same zone models as standard products/charge offers. For example, you can create two promotional products/charge offers, CALL_USA and CALL_INDIA. You can give CALL_USA priority 4 and CALL_INDIA priority 2, while your base charge offer has priority 0. There is no need to create special zone models for the promotional charge offers. If a customer buys the promotional products/charge offers, BRM uses them to rate calls to the US and India. Calls to other countries will be rated using the base product/charge offer.

The capability to use overlay promotions is built into a pipeline, which:

- Recognizes and uses product/charge offer priorities.
- Allows you to associate multiple products/charge offers with the same service and event.
- Allows you to rate calls using overlay promotions only.

> **Note:**
>
> The overlay promotions feature interacts with least cost rating and promotional savings. An EDR can qualify for *either* least cost rating or overlay promotion. It cannot qualify for both.

How Pipeline Modules Process Overlay Promotions

The overlay promotions feature requires no special configuration of pipeline modules. The following modules are involved in the processing of overlay promotions:

- **FCT_CustomerRating**. The module creates a list of associated charge breakdowns and corresponding charge packets ordered from highest to lowest priority. If two products/charge offers have the same priority, the charge offer with the earliest start time is given a higher priority.

  The module then determines whether an EDR qualifies for least cost rating or overlay promotions. If the EDR qualifies for overlay promotions, the module further checks whether the EDR qualifies for promotional savings.

  > **Note:**
  >
  > An EDR can qualify for *either* least cost rating or overlay promotion. It cannot qualify for both. If the EDR qualifies for overlay promotion, it may also qualify for promotional savings.

  If the EDR is eligible for least cost rating, FCT_CustomerRating performs these tasks:

  – Enables least cost rating by setting the DETAIL.CUST_A.LEAST_COST field to **2**.

  – Disables promotional savings and overlay promotion by setting the DETAIL.CUST_A.PROMOTIONAL_SAVING and DETAIL.CUST_A.PROMOTION fields to **1**.

  If the EDR is eligible for overlay promotions and promotional savings, FCT_CustomerRating performs these tasks:

  – Enables promotional savings by setting the DETAIL.CUST_A.PROMOTIONAL_SAVING field to **2**.

  – Disables overlay promotions and least cost rating by setting the DETAIL.CUST_A.PROMOTION and DETAIL.CUST_A.LEAST_COST fields to **1**.

  If the EDR qualifies for overlay promotions but not promotional savings, FCT_CustomerRating performs these tasks:

  – Enables overlay promotions by setting the DETAIL.CUST_A.PROMOTION field to **2.**

  - Disables least cost rating and promotional savings by setting the DETAIL.CUST_A.LEAST_COST and DETAIL.CUST_A.PROMOTIONAL_SAVING fields to **1**.

- **FCT_PreRating**. The module populates the zoning information for each charge breakdown record in the EDR:
  - If least cost rating is used, the module finds the zoning information for all charge breakdown records. See "About Least Cost Rating".
  - If promotional savings or overlay promotion is used, the module finds the zoning information for at least one charge breakdown record. See "About Overlay Promotions".
- **FCT_MainRating**. The module functions differently depending on whether the EDR is rated in least cost rating, overlay promotion, or promotional savings mode:
  - **Least cost mode**. The module rates all charge breakdown records. If it fails to rate any charge breakdown record, it returns an error. See "About Least Cost Rating".
  - **Overlay promotion mode**. The module tries to rate charge breakdown records starting with the highest priority record. If rating is successful for a charge breakdown record, the module selects the charge of that record and populates the INTERN_FOUND_PP_INDEX field with the index of that charge. The module then deletes all other charge breakdown records from the container.
  - **Promotional savings mode**. The module tries to rate charge breakdown records, starting with the highest priority record. If rating is successful for a charge breakdown record, the module selects the charge of that record and populates the INTERN_FOUND_PP_INDEX field with the index of that charge. The module continues rating charge breakdown records until it reaches the last record. It then checks for and selects the last successfully rated record. The module keeps the first and last successful charge breakdown records in the EDR container but deletes all others. See "About Calculating the Promotional Savings".

About Rating with Products and Discounts Whose Validity Starts on First Usage

The effective period of an account's products/charge offers and discounts/discount offers can start when the products/charge offers and discounts/discount offers are first used to rate a subscriber's usage.

If products/charge offers or discounts/discount offers are configured to start on first usage, you set up pipeline rating to set their validity periods when first usage occurs.

> **Note:**
>
> Balance Elements that are granted by products/charge offers and discounts/discount offers can also start on first usage. To set balance element validity when first usage occurs, you configure the batch discounting pipeline. See "About Setting the Validity of Balance Elements Impacted by Discounts ".

When configured to set validity on first usage, the pipeline:

1. Suspends EDRs that use products/charge offers or discounts/discount offers configured to start on first usage until the product/charge offer and discount/discount offer validity periods are set.

   EDRs are not suspended when first-usage balance element validity is set because the pipeline calculates and stores the validity period in memory. The pipeline uses the stored validity period if it needs to consume balance elements for any other events before the balance validity period has been set in the database.

2. Sends the charge offer or discount offer information to a file in the output stream.

   BRM processes the file, sets the validity periods in the database, and charges any applicable purchase and cycle fees. While their validity periods are being set, the products/charge offers and discounts/discount offers are locked.

3. If the event is discounted and the discount balance impact consumes a balance element balance that starts on first usage, the pipeline sends the balance element information to a file in the output stream.

   BRM processes the file and sets the balance element validity periods in the database. If the validity period of all first-usage balance elements are configured to be synchronized, BRM also sets the validity period of those balance elements.

4. Rates the events when the EDRs are recycled.

If the pipeline transaction is rolled back or canceled, the validity period of any charge offer or discount offer that was set in the transaction is unset.

BRM uses the Account Synchronization Data Manager (DM) to synchronize charge offer and discount offer validity periods in the BRM database and pipeline memory.

The following modules are involved in setting the effective periods of products/charge offers and discounts/discount offers:

- The FCT_FirstUsageNotify module. See "About Suspending EDRs for Products and Discounts that Start on First Usage".
- The DAT_AccountBatch module determines if an account's products/charge offers and discounts/discount offers start on first usage, determines the state of the validity period, and calculates the validity periods based on the EDR timestamp. It sends this information to the FCT_Account module.

  The state of a validity period is used for coordinating the validity-setting and EDR-suspension processes. The state can be one of the following:

  - **NOT_FirstUsage**: Indicates that the charge offer or discount offer's validity period is already set in the database.
  - **NEW_FirstUsage**: Indicates that the charge offer or discount offer is configured to start on first usage and that its validity period has not yet been initialized.
  - **INIT_FirstUsage**: Indicates that the charge offer or discount offer's validity period has been initialized but has not yet been stored in the database.

  If the validity periods of all first-usage balance elements in the bundle should be synchronized, DAT_AccountBatch retrieves a list of the balance groups that have first-usage validity and are associated with the products/charge offers or discounts/discount offers in the bundle.
- The FCT_ApplyBalance module. See "About Setting the Validity of Balance Elements Impacted by Discounts ".
- The OUT_GenericStream module and related modules. See "Configuring Pipeline Output for First-Usage Products, Discounts, and Balance Elements".

To set the validity for products/charge offers and discounts/discount offers that start on first usage, you must also perform these tasks:

- Configure BRM recycling to recycle suspended EDRs. For more information, see "Setting Up Recycling for Events whose Product or Discount Validity Starts on First Usage".
- Use Universal Event (UE) Loader to update the validity periods in the BRM database. See "About Updating Validity Period Information in the BRM Database".

About Suspending EDRs for Products and Discounts that Start on First Usage

Use the FCT_FirstUsageNotify module to suspend EDRs while BRM sets the validity periods of products/charge offers and discounts/discount offers that start on first usage. See "FCT_FirstUsageNotify".

> **Note:**
>
> To recycle and rate EDRs that are suspended while validity is being set, you must configure BRM standard recycling. See "Configuring Standard Recycling".

If a call details record (CDR) uses both a charge offer and a discount offer that start on first usage for the first time, the EDR is suspended and recycled twice: once to set the charge offer validity period and again to set the discount offer validity period. This is because the charge offer may grant balance elements that can be impacted by the discount offer, so the charge offer's purchase and cycle events need to be processed before the discount offer is evaluated.

The FCT_FirstUsageNotify module performs the following tasks:

1. Checks whether the products/charge offers and discounts/discount offers used to rate an EDR have validity periods that start on first usage.
2. If a product/charge offer and discount/discount offer starts on first usage, flags the EDR for suspense and recycling by setting the ERR_FIRST_USAGE_VALIDITY_NEEDS_INITIALIZING error code and the **FirstUsageValidity** recycle key in the EDR.
3. Notifies the DAT_AccountBatch module that the validity period of the charge offer or discount offer is being set.
4. Sends the charge offer and discount offer validity information to a separate output stream.
5. Continues to flag for suspense and recycling subsequent EDRs that use the products/charge offers or discounts/discount offers with first-usage validity until the validity period is set.

FCT_FirstUsageNotify predetermines whether the FCT_Reject module will reject an EDR. FCT_FirstUsageNotify does not process EDRs that will be rejected because product/charge offer, discount/discount offer, or balance element validity should not be set for EDRs that will be otherwise suspended.

Configuring Pipeline Output for First-Usage Products, Discounts, and Balance Elements

In the batch rating pipeline, you configure two output streams: one for files containing products/charge offers and discounts/discount offers that start on first usage, and another for files containing balance elements that start on first usage.

To create the first-usage output streams, configure the following:

- The OUT_GenericStream module. See "Configuring First-Usage Output Streams ".
- The DataDescription section of the registry. See "Specifying the First-Usage Format and Mapping Files in the DataDescription Registry".

Configuring First-Usage Output Streams

Configure the OUT_GenericStream module to write the first-usage products/charge offers, discounts/discount offers, and balance elements to an output file.

The default output grammar description files are:

- For products/charge offers and discounts/discount offers, **FirstUsageNotify_OutGrammar.dsc**.
- For balance elements, **FirstUsageResource_OutGrammar.dsc**.

In the EXT_OutFileManager section of the registry, specify the temporary file specifications as follows:

- For first-usage products/charge offers and discounts/discount offers:

```
OutputPath = ./data/out/firstUsage/prod_disc
OutputPrefix = test_PROD
OutputSuffix = .out_1
TempPrefix = .
TempDataPath    = ./data/out/firstUsage/prod_disc
TempDataPrefix  = prod.tmp.
TempDataSuffix  = .data_1
```

- For first-usage balance elements:

```
OutputPath = ./data/out/firstUsage/resources
OutputPrefix = test_RES
OutputSuffix = .out_1
TempPrefix = .
TempDataPath    = ./data/out/firstUsage/resources
TempDataPrefix  = res.tmp.
TempDataSuffix  = .data_1
```

See "OUT_GenericStream".

Specifying the First-Usage Format and Mapping Files in the DataDescription Registry

Configure the first-usage stream formats, input mapping, and output mapping in the batch pipeline DataDescription registry section as follows:

```
DataDescription
{
   Standard
   {
```

```
   ModuleName = Standard
   Module
   {
   StreamFormats
   {
      FIRST_USAGE_NOTIFY_OUTPUT=./formatDesc/Formats/FirstUsageNotify/FirstUsageNotify.dsc
      FIRST_USAGE_RESOURCES=./formatDesc/Formats/FirstUsageNotify/FirstUsageResource.dsc
   }
   . . .
   OutputMapping
   {
      FIRST_USAGE_PROD_DISC=./formatDesc/Formats/FirstUsageNotify/FirstUsageNotify_OutMap.dsc
      FIRST_USAGE_RESOURCES=./formatDesc/Formats/FirstUsageNotify/FirstUsageResource_OutMap.dsc

   }
   }
   }
}
```

For more information, see "Configuring Output for Rated Events".

About Updating Validity Period Information in the BRM Database

You use UE Loader to update validity periods in the BRM database when first usage occurs.

You use the ConfigurableValidityHandler batch handler to run the UE Loader utility (**uel**).

Perform the following tasks to update validity periods for first-usage products/charge offers, discounts/discount offers, and balance elements:

- Load the first-usage validity Universal Event Mapper templates. See "Loading the First-Usage Validity Templates".
- Configure the ConfigurableValidityHandler batch handler. See "Configuring the ConfigurableValidityHandler Batch Handler".
- Configure Batch Controller. See "Configuring Batch Controller to Start the ConfigurableValidityHandler Batch Handler".

Loading the First-Usage Validity Templates

BRM provides two Universal Event (UE) Mapper event import templates that are used by UE Loader to process the first-usage product/charge offer, discount/discount offer, and balance element files output by the OUT_GenericStream module:

- **FirstUsageProductsDiscounts.xml**: This template defines the format of the first-usage product/charge offer and discount files.
- **FirstUsageResources.xml**: This template defines the format of the first-usage balance element files.

You must load these templates into the BRM database by running the **pin_uei_deploy** utility.

For example:

```
pin_uei_deploy -c -t FirstUsageProductsDiscounts -i BRM_home/formatDesc/Formats/
FirstUsageNotify/FirstUsageProductsDiscounts.xml

pin_uei_deploy -c -t FirstUsageResources -i BRM_home/formatDesc/Formats/FirstUsageNotify/
FirstUsageResources.xml
```

Configuring the ConfigurableValidityHandler Batch Handler

BRM provides the ConfigurableValidityHandler batch handler for loading first-usage validity data. You must configure ConfigurableValidityHandler to run the following utilities:

- **pin_rel:** This utility loads data for events that are rated in batches.
- **uel**: This utility loads validity period data for products/charge offers, discounts/discount offers, and balance elements when they are used for the first time.
- **pin_load_rerate_jobs**: You must also configure to run this utility if you configured Pipeline Manager to detect and rerate events that are rated out of order. This utility creates rerate jobs for the events that were rated out of order.

For more information, see "About Using a Single Batch Handler to Run Multiple Loading Utilities".

To configure the ConfigurableValidityHandler batch handler, you edit the handler's configuration file (*BRM_home***/apps/pin_rel/ConfigurableValidityHandler_config.values**). You specify handler, processing, and staging directories for each loading utility that this batch handler runs.

Configuring Batch Controller to Start the ConfigurableValidityHandler Batch Handler

Batch Controller polls the pipeline output directories and starts the ConfigurableValidityHandler batch handler when a data file is ready to be loaded.

> **Note:**
>
> If you use the ConfigurableValidityHandler batch handler, do not configure Batch Controller to run a separate instance of the handlers that load pipeline batch-rated events, out-of-order rerating requests, or first-usage validity data. (You may configure a separate instance of the Rated Event (RE) Loader handler that loads suspended events into **/suspended_usage** objects.) If you have already configured Batch Controller to run the out-of-order event handler (OODHandler), remove those entries or comment them out.

To configure Batch Controller to start the ConfigurableValidityHandler batch handler:

1. Open the Batch Controller **Infranet.properties** file in *BRM_home***/apps/batch_controller**.
2. Edit the file to include entries for the ConfigurableValidityHandler batch handler.

> **Note:**
>
> - If you have already configured the Batch Controller **Infranet.properties** file for RE Loader, you can use the existing RE Loader entries and most of their values or you can change them. Entries that you must change are marked with an asterisk (*).
> - If you have not yet configured the Batch Controller **Infranet.properties** file for RE Loader, you must add ConfigurableValidityHandler values for all of the following entries.

Table 2-2 lists the entries you must set and the default values used for RE Loader:

Table 2-2 Entries for RE Loader

Entry	Description
batch.random.events	Identifies this specific configuration for triggering Batch Controller. For example: `batch.random.events CdrFileEvent`
event_name.**name**	(Optional) A name for the configuration identifier. For example: `CdrFileEvent.name CdrFileEvent`
event_name.**handlers**	The batch handler identifier. For example: `CdrFileEvent.handlers ConfigurableValidityHandler` The default is **relHandler**.
event_name.**file.location**	The full path to the pipeline output directory where the rated-event files are deposited. For example: `CdrFileEvent.file.location /export/portal/integRate`
event_name.**file.pattern**	The rated-event output file name. You can use an asterisk (*****) to represent zero or more characters in the file name. No other wildcards (metacharacters) are supported. For example: `CdrFileEvent.file.pattern cdr*.dat`
***handler_name*.name**	The name of the batch handler that is run. For example: `ConfigurableValidityHandler.name ConfigurableValidityHandler` **Note:** If you did not change the default RE Loader value for *event_name.handlers*, this entry should be: `relHandler.name ConfigurableValidityHandler`
handler_name.**max.at.lowload.time** *handler_name*.**max.at.highload.time**	The number of batch handler instances that can run concurrently during periods of low load and high load usage. Typical default settings are **6** at low load and **3** at high load. For example: `ConfigurableValidityHandler.max.at.highload.time 3` `ConfigurableValidityHandler.max.at.lowload.time 6`
***handler_name*.start.string**	The full path to the ConfigurableValidityHandler handler. For example: `ConfigurableValidityHandler.start.string` *`BRM_home`*`/apps/pin_rel/ConfigurableValidityHandler.pl` **Note:** If you did not change the default RE Loader value for event_name.handlers, this entry should be: `relHandler.start.string` *`BRM_home`*`/apps/pin_rel/ConfigurableValidityHandler.pl`

3. Save and close the file.

4. Stop and restart Batch Controller.

Setting Up Recycling for Events whose Product or Discount Validity Starts on First Usage

To suspend and recycle EDRs while product/charge offer and discount/discount offer validity is being set, you must configure BRM standard recycling. See "Configuring Standard Recycling".

For information about standard recycling, see "About Standard Recycling".

To recycle events suspended due to first-usage validity, you run the **pin_recycle** utility. Use the **-k** parameter and specify the **FirstUsageValidity** recycle key. For example:

```
pin_recycle -k FirstUsageValidity
```

For more information, see "Using Standard Recycling to Recycle Suspended EDRs" and "pin_recycle".

About First-Usage Validity for Events Rated Out of Order

In the batch rating pipeline, if events are rated in a different order than they occurred, validity periods that are based on first usage may be incorrectly set. This can happen if the first event rated, which initiated the validity period, wasn't actually the first usage event. To correct the validity periods, you must rerate the events. Rerating corrects the order of the events and resets the validity period based on the actual first-usage event.

About Customer Rating

Customer rating assigns a charge to an EDR based on customer data. The FCT_CustomerRating module performs customer rating. The module creates an associated charge breakdown record and one charge packet, which includes the charge code. The FCT_MainRating module uses the charge code to rate the event.

To assign the charge, the FCT_CustomerRating module does the following:

- If the service that generated the event includes a service-level charge extended rating attribute (ERA), that charge is used.

  If you're using subscription services, and a subscription service and member service both own a service-level charge ERA, the member service's ERA has priority and is used for selecting the charge.
- If there is no service-level charge ERA, but there is an account-level charge ERA, that charge is used.
- If there is no charge ERA, the charge associated with the last product/charge offer found is used when there is more than one product/charge offer available.

To configure customer rating, see "FCT_CustomerRating".

Assigning a Default Charge and Default Segment for Customer Rating

When you configure the FCT_CustomerRating module, you can assign a default charge and default segment to use if no customer information for the A number is found.

- Use the **DefaultRateplan** entry to specify the default charge name in case there is not enough information to assign a charge. You can change this value by using a semaphore.

- Use the **DefaultSegment** entry to specify if segment rating is used, the default segment to use if no segment is found. You can change this value by using a semaphore.

See "FCT_CustomerRating".

About Customer Rating and Service Level Agreements

You define service-level agreement (SLA) codes in Pricing Center or Pipeline Configuration Center (PCC) to map service level agreements (SLAs) to a usage-scenario (USC) group, rate-service class (RSC) group, and rule set. The SLA mapping is stored in the IFW_SLA database table.

During customer rating, if the FCT_CustomerRating module finds an SLA code, the module looks up the code in the IFW_SLA database table and adds the following data to the charge packet:

- RSC group
- USC group
- Rule set

Pipeline Manager can use any one of these to determine the rate for the service level usage:

- The RSC group is used by the FCT_RSC_Map module to find the RSC map. FCT_RSC_Map maps the usage class, usage type, service code, and impact category to a new service class. See "About Setting Up RSC Mapping".
- The USC group is used by the FCT_USC_Map module to find the USC map. FCT_USC_Map maps the usage class, usage type, service code, service class, and zone to a new usage type and impact category.

About Rate-Service Class Mapping

You map a rate-service class (RSC) to perform rating based on the quality of service (QoS) when you set up service-level agreements (SLAs). RSC mapping is performed by the FCT_RSC_Map module, which maps the usage class, usage type, service code, and impact category to a new service class.

You create RSC map groups to provide a different service class for each level of service quality. You link the RSC map group to an SLA code in Pricing Center or Pipeline Configuration Center (PCC). The FCT_CustomerRating module looks up the SLA code and adds the associated RSC group to the charge packet (see "About Customer Rating and Service Level Agreements"). FCT_RSC_Map uses the RSC group to evaluate the EDR and find the correct RSC map.

The RSC map group is specified in the INTERN_SLA_RSC_GROUP field of the EDR. This field is filled in by FCT_CustomerRating when the product/charge offer includes a service-level agreement ERA. If INTERN_SLA_RSC_GROUP is empty, the default RSC group is used. You specify the default RSC group in the **DefaultRscGroup** entry of the FCT_RSC_Map registry.

About Setting Up RSC Mapping

To set up RSC mapping, do the following:

1. Use Pricing Center or Pipeline Configuration Center (PCC) to create RSC maps and RSC map groups.
2. Use Pricing Center or PCC to create SLA codes and link them to RSC map groups.

3. Configure FCT_RSC_Map. See "FCT_RSC_Map".

About RSC Maps

To assign a new service class to the EDR, FCT_RSC_Map reads data from the EDR and evaluates it according to the RSC map. An RSC map includes one or more mapping rules that specify the data that must be matched to apply the new service class.

When you create RSC maps, you create a mapping rule for each new service class. You can also define the order in which the rules are evaluated. The new service class is derived from the first rule that matches the data. If no matching rule is found, FCT_RSC_Map uses the default RSC map as defined in the registry. If no default value exists, no mapping is performed.

You can use the following EDR data to create an RSC mapping rule:

- The charge used to rate the EDR
- The QoS requested
- Usage class
- Usage type
- Service code
- Service class
- Impact category

When you create the mapping rules, you can use regular expressions. For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

To create a valid mapping, the data in the EDR must match with all of the mapping data.

About Main Rating

The FCT_MainRating module carries out the pipeline rating functionality.

When an EDR is ready for rating, it includes all the data needed by FCT_MainRating; for example, the service class, usage class, zone, and charge. The module uses the charge to rate the EDR. To rate the EDR, the module uses information about dates, times, and the pricing to apply charges. When rating is finished, the EDR contains complete charge breakdown data, including charge packets with charges.

FCT_MainRating uses criteria in the charge configuration that apply to an EDR to find the correct pricing to use for rating. If a single event is rated by using different time periods, such as peak and off-peak, more than one pricing can be used. If an event is mapped to a price selector, the model selector's rules are evaluated to choose a pricing.

If the charge configuration includes an alternative pricing, the module creates a new charge packet. The EDR is rated again by using the alternative pricing. The charge packet is flagged with the A (Alternative) pricing type.

During main rating, the module checks for the RUM group associated with the service code. At least one charge packet is added to the EDR for each RUM assigned to the RUM group. A single charge packet is generated for each time period, RUM, and balance element that is used. (The balance elements are defined in the pricing.)

To configure main rating, see "FCT_MainRating".

About Rate Adjustment

You use rate adjustments to provide discounts based on date, time, service, and other event attributes.

To set up rate adjustment, do the following:

1. Create rules that specify which EDRs to adjust and how to adjust them. You have two options:
   - Create rate adjustment rules in Pricing Center or Pipeline Configuration Center (PCC). In this case, the rate adjustment data is stored in the Pipeline Manager database.
   - Create a file that defines usage scenario maps. In this case, the FCT_RateAdjust module reads the file. For information on creating the file, see "Creating a Rate Adjustment Rules File".
2. Configure FCT_RateAdjust. See "FCT_RateAdjust".

Creating a Rate Adjustment Rules File

The rate adjustment rules can be defined in an ASCII file:

- Each rule consists of a list of fields. The following table describes the meaning of each field.
- Every new line defines an adjustment rule.
- Fields are separated by semicolons (;).
- Comment lines start with #.
- Empty lines are allowed.

> **Note:**
>
> The value of the field rank is ignored. The evaluation order of the rules is given by the order of the rules within the file.

Table 2-3 lists the fields in the file:

Table 2-3 Fields in a Rate Adjustment File

Position	Field	Description
1	Rank	Specifies the evaluation order of the rules. This is ignored because the evaluation order is specified within the file.
2	Rate plan	Specifies the charge to adjust.
3	Rate plan version	Specifies the charge version.

Table 2-3 (Cont.) Fields in a Rate Adjustment File

Position	Field	Description
4	Valid from	Specifies the start date for the rate adjustment. This can be either a date with the format *YYYYMMDD* or a weekday with an optional timestamp; for example: • 19990524 • SAT • MON16:00 If the field is left empty, the earliest possible date (19010101) is used.
5	Valid to	Specifies the end date for the rate adjustment. This can be either a date with the format *YYYYMMDD* or a weekday with an optional timestamp. If the field is left empty, the latest possible date (20370205) is used.
6	Time from	Specifies the start time for the rate adjustment. The format is *HH*:*MM*. The default is **00:00**.
7	Time to	Specifies the end time for the rate adjustment. The format is *HH*:*MM*. Example: To set up a discount that is valid on weekends between 13:00 and 14:00, you have to set the following: • **ValidFrom** = SAT • **ValidTo** = SUN • **TimeFrom** = 13:00 • **TimeTo** = 14:00
8	Quantity value	Specifies the maximum quantity value for an EDR container. If this maximum is exceeded, the mapping rule will not be used. If this field is left empty or if a 0 is specified, the rule is valid for every quantity value. Example: You can use this entry to avoid discounting for calls longer than 120 seconds by setting the field to 120.
9	Usage class	Specifies the compare pattern for the usage class.
10	Usage type	Specifies the compare pattern for the usage type.
11	Service code	Specifies the compare pattern for the service code.
12	Service class	Specifies the compare pattern for the service class.
13	Impact category	Specifies the compare pattern for the impact category.
14	Source network	Specifies the compare pattern for the source network.
15	Destination network	Specifies the compare pattern for the destination network.
16	Discount type	Specifies the discount type.
17	Discount value	Specifies the discount value.
18	Comment	Specifies the rate adjustment name.

About Consolidation for BRM Billing

The FCT_BillingRecord module consolidates charge packets and discount packets into an associated BRM billing record in the EDR. This data is loaded as a rated event by RE Loader.

The associated BRM billing record includes the POIDs of the **/account** object and the **/service** object and the POID of the item that receives the balance impact. If an event affects more than one customer balance, an associated BRM billing record is created for each balance.

An associated BRM billing record can contain one or more *balance impact packets*. The data in a balance impact packet is loaded into an **/event** object balance array. Therefore, the data includes information about the charged amount, the charge, and balance elements.

The balance impact packet also includes data in the PIN_INFO_STRING field. This field contains the information about the individual charge packets.

Each balance impact packet includes data for one balance impact per balance element. If there are different G/L IDs for the same balance element, a balance impact packet is created for each G/L ID.

To configure FCT_BillingRecord, see "FCT_BillingRecord".

> **Note:**
>
> Don't use FCT_BillingRecord in a CIBER roaming revenue assurance environment. For more information, see "Billing Consolidation with CIBER Roaming and Revenue Assurance".

How the FCT_BillingRecord Module Works

FCT_BillingRecord uses the data in the EDR to determine if the charges should be included in the associated BRM billing record. To do so, the module checks the following data. If any of these do not match, no associated billing record is created.

- The record type must be 981. This record type is created by the FCT_CustomerRating module for customer rating.
- The charge packet must use the following:
  - The standard pricing type: PRICEMODEL_TYPE = S
  - A retail charge: RATEPLAN_TYPE = R
  - A currency type that matches the currency type specified in the FCT_BillingRecord module registry. The options are Home, Billing, and Rating.
  - A currency that matches one of the entries specified in the FCT_BillingRecord module registry. See "FCT_BillingRecord".

If the charge packet meets the criteria, the module sums the amount, discount, and quantity for each BRM balance element and creates the associated BRM billing record and one or more balance impact packets.

For balance monitoring, FCT_BillingRecord generates a monitor packet for each monitor group.

To get data, FCT_BillingRecord connects to the following data modules:

- The DAT_AccountBatch module, which provides data about items.
- The DAT_ItemAssign module, which provides data about items assigned for sponsorship events.
- The DAT_Currency module, which provides data for converting currency symbols to numeric values.
- The DAT_BalanceBatch module, which provides the **ObjectCacheType** value from the **/balance_group** object.

Billing Consolidation with CIBER Roaming and Revenue Assurance

For CIBER roaming and revenue assurance, use the ISC_PostRating iScript instead of FCT_BillingRecord.

ISC_PostRating adds all the retail and wholesale charges and puts them in the DETAIL.RETAIL_CHARGED_AMOUNT_VALUE and DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE fields.

> **Note:**
>
> ISC_PostRating and FCT_BillingRecord perform similar billing consolidation, but ISC_PostRating doesn't load balance impact data into the database.

See "ISC_PostRating".

How the ISC_PostRating iScript Works

ISC_PostRating sets the following attributes:

- Wholesale and retail impact category
- Charged amount value
- Amount currency
- Tax treatment

for an EDR based on these values specified in the Pipeline Manager registry:

- Balance Element type
- Pricing type
- Currency type

> **Note:**
>
> This iScript accesses only EDR fields. It doesn't access the database.

Adding Pipeline Rating Data to an Invoice

When you rate usage by using pipeline rating, information about how events are rated are stored in the EDR. You can display this information on invoices. For example, if a call spans

two rates, for peak and off-peak time, you can display the rate used for each part of the call. For example:

Table 2-4 Displaying Pipeline Rating Data

Date/time	Called number	Duration	Average rate per unit	Total charge
12/12/2003 1200	4085551212	10min	$0.125	$1.25
12/12/2003 1200	4085551212	5min	$0.15	$0.75
12/12/2003 1200	4085551212	5min	$0.10	$0.50

In the example shown in Table 2-4, the data is stored in the INTERN_PRICE_MDL_STEP_INFO EDR field.

To add pipeline rating invoice data to your invoices, you need to configure the OUT_GenericStream module **AddInvoiceData** registry entry to add the data to the BRM billing record.

Specifying Invoice Data from Pipeline Manager and Custom Applications

If you process events that originate in Pipeline Manager or a custom application, you can create a template to specify the event invoice data so that it can be included in your invoices.

BRM provides a default template file (**pin_invoice_data_map**) located in *BRM_home***/sys/data/config**. This file includes a default **INTEGRATE** template for Pipeline Manager invoice data. You can modify the **INTEGRATE** template or add new templates to the file.

To use this feature, you must do the following:

- Load the new invoice data template. See "Loading the Invoice Data Map Templates".
- Enable event caching in the CM configuration file. See "Enabling Event Caching".
- Configure the output module to add invoice data. See "Adding Invoice Data to Pipeline Output".

Using Data Map Templates

Templates in the **pin_invoice_data_map** file define the fields in invoice data records. If you use different invoice record formats, you can create a template for each record format. The data map file can include any number of templates.

When you define templates, you specify the BRM flist fields that map to the invoice record fields. When the invoice data is processed, the fields are passed in an flist to the invoicing opcodes for processing.

The default **INTEGRATE** template defines Pipeline Manager invoice data. You can modify the **INTEGRATE** template or create new templates.

To create or modify templates in the **pin_invoice_data_map** file:

1. Open the *BRM_home***/data/config/pin_invoice_data_map** file.
2. Change the **INTEGRATE** template or add a new template to the end of the file. Use the following syntax:

```
ID template_name
field_level   field_name
field_level   field_name
field_level   field_name
...
```

where:

- *template_name* is the name of the template.
- *field_level* is the level of the field in the flist.
- *field_name* is the associated BRM field.

For example:

```
ID INTEGRATE
0 PIN_FLD_CALLING_NUMBER
0 PIN_FLD_CALLED_NUMBER
. . .
0 PIN_FLD_BAL_IMPACTS
1 PIN_FLD_RATE_TAG
1 PIN_FLD_AMOUNT
```

3. Save and close the file.

The order of the fields in the template must correspond to the order of the fields in the invoice record. Any custom invoice data records that you process must follow these rules:

- Fields must be separated by a pound symbol: #
- Arrays and substructs must be enclosed in angle brackets: **< >**
- Each balance impact element must end with a pipe symbol: **|**

> **Note:**
>
> The pipe is optional after the last element.

- The first field in the record must be the name of the corresponding invoice data template and be preceded by the @ symbol.

  For example: **@INTEGRATE**

- The template name in the record must match the template name in the **load_pin_invoice_data_map** file.

Loading the Invoice Data Map Templates

After defining invoice data map templates, load them into the BRM database.

> **Note:**
>
> When you run **load_pin_invoice_data_map**, it overwrites the existing invoice data templates. If you are updating a set of templates, you cannot load new templates only. You must load complete sets of invoice data templates each time you run **load_pin_invoice_data_map**.

> **Note:**
>
> To connect to the BRM database, **load_pin_invoice_data_map** needs a configuration file in the directory from which you run the utility.
>
> If you defined a custom field for invoicing, you must add the full path name of the mapping file to the **load_pin_invoice_data_map** utility's **pin.conf** file.

To load the data map templates:

1. Go to the directory that contains the utility's configuration file.
2. Run the following command, which loads the data map template:

```
load_pin_invoice_data_map -d -v pin_invoice_data_map
```

Enabling Event Caching

Pipeline Manager caches invoice data before sending it to BRM for processing. Therefore, you must enable event caching to include Pipeline Manager invoice data in your invoices.

To enable event caching:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Set the **event_cache** entry to **1**:

```
- fm_inv event_cache 1
```

3. Save and close the file.

Adding Invoice Data to Pipeline Output

To add data from a pipeline to your invoices, configure the OUT_GenericStream module to add the data to the BRM billing record.

To output invoice data, add the following registry parameter to all OUT_GenericStream modules:

```
AddInvoiceData = TRUE
```

This parameter is read by the output grammar. When set to **TRUE**, the output module adds invoice data to each BRM billing record.

Using the pin_virtual_time Utility with Pipeline Manager

If you use **pin_virtual_time** to test charging and billing, you need to set the **VirtualTime** parameter for the DAT_BalanceBatch module to **True** to ensure that balances are calculated correctly.

3
Configuring EDR Input Processing

This chapter describes how to set up input processing for the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

About the Input Process

To process incoming data, Pipeline Manager input modules convert data into an internal EDR format understood by the pipeline function modules. The input data is contained in files, such as CDRs for telco rating.

The input process works as follows:

1. Your mediation system automatically places CDR files into a directory.

   If you start a pipeline and the directory already includes input files, they are processed according to the last-modified timestamp.

2. The input module uses the stream format description file to create separate records from the data. For example, the data is separated into HEADER records, DETAIL records, and TRAILER records. DETAIL records can include data for one service only (for example, GSM *or* GPRS).

3. The input module uses the input grammar file to process each record. The module verifies that the syntactical order for each record is correct. For example, if a particular field is supposed to include 10 characters, the input module uses the grammar file to check that. It also uses the grammar file to normalize data, such as the A number.

   If an error is found, processing stops and an error message is logged.

4. The input module creates an EDR container for the data. See "About EDRs".

5. The input module uses an input mapping file to copy data from the external file into the appropriate EDR container fields.

   > **Note:**
   >
   > A separate input mapping file is required to process each external data format. See "Setting Up an Input Mapping File".

6. The input module puts the EDR into the input buffer for processing by the function modules.

> **Note:**
>
> A separate input grammar file is required to process each external data format. See "Setting Up an Input Grammar File".

The following examples show how the A number in a CDR is mapped to a rating EDR container.

> **Note:**
>
> These examples show only selected portions of the stream format description, grammar, and mapping files.

The following example shows part of a stream format description file. This section of the file describes how a DETAIL record is formatted and lists the fields in the record (fields are separated by a semicolon). The first field (SERVICE) stores the service code, the second (A_NUMBER) stores the A number, and so forth. Data in each of the fields must be of type **AscString()**.

```
DETAIL(SEPARATED)
{
Info
{
Pattern = ".*\n";
FieldSeparator = ';';
RecordSeparator = '\n';
}
SERVICE     AscString();
A_NUMBER    AscString();
B_NUMBER    AscString();
...
```

The next example shows part of an input grammar file. The first two lines create a new block of data in an EDR container (**edrNew**) and specify the location in an external file from which the data should be copied (**edrInputMap**). Inside the new block, the next two lines normalize the A number and then add a field to the EDR container called DETAIL.A_NUMBER.

```
edrNew( DETAIL, CONTAINER_DETAIL );
edrInputMap( "SAMPLE.DETAIL.STD_MAPPING" );
...
number = normalizeNumber( edrString( DETAIL.A_NUMBER ), "00", 0 );
edrString( DETAIL.A_NUMBER ) = number;
```

The next example shows part of an input mapping file. In this example, the A_NUMBER defined in the stream format description example is mapped to the DETAIL.A_NUMBER field defined in the input grammar example. Note how the nested blocks of data correspond to the **edrInputMap** entry (**SAMPLE.DETAIL.STD_MAPPING**) in the input grammar example.

```
SAMPLE
  DETAIL
  {
    HDR_MAPPING
    {
    "010"       -> HEADER.RECORD_TYPE;
    ...
  }
  TRL_MAPPING
  {
    "090"       -> TRAILER.RECORD_TYPE;
    ...
  }

  STD_MAPPING
```

```
  {
    "020"       -> DETAIL.RECORD_TYPE;
    0           -> DETAIL.DISCARDING;
    "00"        -> DETAIL.A_MODIFICATION_INDICATOR;
    0           -> DETAIL.A_TYPE_OF_NUMBER;
    "0"         -> DETAIL.A_NUMBERING_PLAN;
    A_NUMBER    -> DETAIL.A_NUMBER;
...
```

You can map one field in an external file to multiple fields in an EDR container. The following example shows part of the same input mapping file. In this part of the file, however, the data block for the GSM service maps the A_NUMBER to a different field:

```
GSMW_MAPPING
  {
  "520"       -> DETAIL.ASS_GSMW_EXT.RECORD_TYPE;
  A_NUMBER    -> DETAIL.ASS_GSMW_EXT.A_NUMBER_USER;
  B_NUMBER    -> DETAIL.ASS_GSMW_EXT.DIALED_DIGITS;
...
  }
```

About Setting Up Input Processing

To set up input processing, do the following:

1. **(CDR processing only)** Define a CDR file input directory in your pipelines. Configure your mediation system to put the files in this folder automatically.

   > **Note:**
   >
   > You can configure your system to route CDR files from a single input directory to multiple identical pipelines.

2. Set up a stream format description file. You can start with the sample files that are provided. See "Creating a Stream Format Description File".
3. Set up an input mapping file. You can start with the sample files that are provided. See "Setting Up an Input Mapping File".
4. If necessary, set up an input grammar file. In most cases, you do not need to modify the default grammar file. If you modify an EDR container (for example, if you add fields), you might need to modify the input grammar to ensure that data in your EDR containers is correctly formatted. See "Setting Up an Input Grammar File".

   > **Note:**
   >
   > If you customize an EDR container description, you must ensure that your customizations do not affect existing module functionality. Many modules require data from default EDR fields.

5. Configure these input sections in the registry:
   - The **DataDescription** section. See "Configuring the Input DataDescription Registry Section".
   - The **InputBuffer** section in the **Pipeline** section.

- The **Input** section. See "Configuring the Input Section in the Registry".

The sample Pipeline Manager registry files include stream format description, input mapping, and input grammar files that convert data using the rating EDR and TAP formats.

About Input Processing File Types

Input processing uses the following types of files:

- **Input file**: The file from the external system.
- **Temporary file**: The same file as the input file, but renamed as a temporary file during processing. If the file is rejected, this file is not used.
- **Done File**: The same file as the input file, but renamed as a done file after processing has been successfully completed.
- **Error File**: The same file as the input file, but renamed as an error file if the file is rejected.

These files are managed by the EXT_InFileManager module. See "About Getting Pipeline Input from Files".

Creating a Stream Format Description File

To create a stream format description file, first identify the data in your external files that you need to use for rating. You can then either start with one of the sample files or create your own.

In the following example, the external file is a CDR. It includes three records: a HEADER, a TRAILER, and a DETAIL. Each record starts with a character that identifies its EDR container content type (**H** for HEADER, **T** for TRAILER, and **D** for DETAIL), and each record has a fixed structure.

- The example HEADER record in Table 3-1 has two fields:

Table 3-1 Header Record in Stream Format Description

Field	Length	Description
IDENTIFIER	1	The character **H**.
CREATION_TIME	14	The creation time of the CDR stream in the format *YYYYMMDDHHMMSS*.

- The example DETAIL record in Table 3-2 has five fields:

Table 3-2 Detail Record in Stream Format Description

Field	Length	Description
IDENTIFIER	1	The character **D**.
CALLING_PARTY	15	The A number.
CALLED_PARTY	15	The B number.
START_TIMESTAMP	14	The start time of the call in format YYYYMMDDHHMMSS.
DURATION	9	The duration of the call in seconds.

- The example TRAILER record in Table 3-3 has two fields:

Table 3-3 Trailer Record in Stream Format Description

Field	Length	Description
IDENTIFIER	1	The character **T**.
NUMBER_OF_DETAILS	9	The total number of DETAIL records in the stream.

A sample CDR input stream might be the following:

```
H20010613123410D4943311217 4957641506 2001061310011200000045D494106136432 49401531224
2001061310021500000056T000000002
```

The **INP_GenericStream** input module uses the stream format description file to break the CDR input stream into the following records:

```
H20010613123410

D4943311217 4957641506 200106131001120000000045

D494106136432 49401531224 200106131002150000000056

T000000002
```

The input module uses regular expressions to find each record. Table 3-4 lists the three record types and their regular expressions:

Table 3-4 Regular Expressions for the Records

Record	Regular expression	Description
HEADER	"H.{14}"	H followed by 14 arbitrary characters.
DETAIL	"D.{53}"	D followed by 53 arbitrary characters.
TRAILER	"T.{9}"	T followed by 9 arbitrary characters.

The description of the record must contain the following:

- The regular expression used by the input module to recognize the physical record
- The position and types of the fields inside the physical record

A sample format stream description for this example looks like this:

```
SampleFormat
{
  Header(FIX)
  {
    Info
    {
      Pattern = "H.{14}";
    }
    IDENTIFIER        AscString(1);
    CREATION_TIME     AscDate("YYYYmmddHHMMSS");
  }

  Detail(FIX)
  {
    Info
    {
      Pattern = "D.{53}";
    }
```

```
    IDENTIFIER        AscString(1);
    CALLING_PARTY     AscString(15);
    CALLED_PARTY      AscString(15);
    START_TIMESTAMP   AscDate("YYYYmmddHHMMSS");
    DURATION          AscInteger(9);
  }

  Trailer(FIX)
  {
    Info
    {
    Pattern = "T.{9}";
  }

    IDENTIFIER        AscString(1);
    NUMBER_OF_DETAILS AscInteger(9);
  }
}
```

Each field in the record is defined by the field type and value. For example, the fields in the DETAIL record are defined as follows:

```
    IDENTIFIER        AscString(1);
    CALLING_PARTY     AscString(15);
    CALLED_PARTY      AscString(15);
    START_TIMESTAMP   AscDate("YYYYMMDDHHMMSS");
    DURATION          AscInteger(9);
```

Record Types

The **INP_GenericStream** input module uses regular expressions to recognize the data records in the input stream. Different types of data records define fields in different ways:

- Fields can defined by fixed-widths.
- Fields can be separated by special characters.
- The length of the field can be included in the input data (as in ASN.1 input).

Therefore, each data record has a record type that tells the input module how to split up the record into fields. Each data record has an **Info** block in its definition that contains some setup parameters for the record, for example, the field separator for a separated record. The record type determines which parameters can be used.

For example, this DETAIL record uses the record type SEPARATED:

```
DETAIL(SEPARATED)
{
Info
{
Pattern = ".*\n";
FieldSeparator = ';';
RecordSeparator = '\n';
}
SERVICE     AscString();
A_NUMBER    AscString();
B_NUMBER    AscString();
...
```

Record Type SEPARATED

The record type SEPARATED is used for records in which fields are separated by a special field delimiter character. The record itself can be terminated by another character (for example, the end-of-line symbol **\n**). Because there is no length information for the record, the regular expression specified as a pattern must match the full record, including the record separator.

There are no restrictions for the data types that can be used inside the SEPARATED record, although it makes no sense to use binary data types inside this record. The length information calculated from the position of the field delimiters overwrites length information specified in the data types. See Table 3-5.

Table 3-5 Parameters in SEPARATED

Parameter	Value	Description	Mandatory
Pattern	String	Regular expression that defines the entire record. This includes all records and the record separator character.	Yes
FieldSeparator	Character	Character that delimits single fields. Default = Comma (,)	No
RecordSeparator	Character	Character that delimits records. Default = No record separator	No

Record Type FIX

This record type is used for records with predefined width for each field. The record must contain all the fields. The record length is calculated as a sum of the widths of individual fields. Only data types with width information can be used. See Table 3-6.

Table 3-6 Parameters in FIX

Parameter	Value	Description	Mandatory
Pattern	String	Regular expression that identifies the record.	Yes

Record Type ASN

This record type is used for file formats defined in ASN.1, for example, TAP. You can use only the TAP and ASN data types in this record type. See Table 3-7.

Table 3-7 Parameters in ASN

Parameter	Value	Mandatory
Application	Integer	No
Context	Integer	No
Private	Integer	No
Universal	Integer	No

Syntax of the Stream Format Description File

The stream format description file is a simple ASCII file. The following grammar defines the syntax of the stream format description file:

```
<format-description-file> ::=  (<use_directive> | <stream-format>)*
<boolean> ::=                  "true" | "false"
<character> ::=                "'" "single character" "'"
<decimal> ::=                  "0..9"* "." "0..9"+
<extension-field-type> ::=     "any field type defined in a user extension"
<extension-record-type> ::=    "any record type defined in a user extension"
<field-name> ::=               <identifier>
<field-type> ::=               "AscString" | "AscDecimal" | "AscLong" | "AscDate" | ... |
                               <extension-field-type>
<format-name> ::=              <identifier>
<identifier> ::=               "a..z,A..Z,_" "a..z,A..Z,0..9,_"*
<info-block> ::=               "Info" "{" <info-parameter>* "}"
<info-parameter> ::=           <identifier> "=" <value> ";"
<integer> ::=                  "0..9"+
<record-definition> ::=        <record-name> "(" <record-type> ")" "{" <info-block>
                               <record-field>* "}"
<record-field> ::=             <field-name> <field-type> "(" [<field-parameter>
                               [, <field-parameter>]*] ")" ";"
<record-name> ::=              <identifier>
<record-type> ::=              "FIX" | "SEPARATED" | "ASN" | <extension-record-type>
<stream-format> ::=            <format-name> "{" <record-definition>* "}"
<string> ::=                   "\"" "any character"* "\""
<use_directive> ::=            "use" <identifier> ";"
<value> ::=                    <decimal> | <integer> | <identifier> | <string> |
                               <character> | <boolean>
```

Supported Data Types for the Stream Format Description File

Each entry in the stream format description file assigns a data type to a field. For example, this line assigns the **AscString** data type to the A number:

```
CALLING_PARTY     AscString(15);
```

Pipeline rating supports the following categories of data types:

- ASCII Data Types
- ASN.1 Data Types
- TAP Data Types

ASCII Data Types

Pipeline Manager supports the following ASCII data types:

- AscDate
- AscDecimal
- AscInteger
- AscString
- AscRawString

AscDate

Use the **AscDate** data type to read and write date/time information as an ASCII string. The **AscDate** data type can be used without any parameters or with a string specifying the used date format.

```
AscDate( [String format] )
```

The format string uses the following patterns:

- **%Y**: The year including the century (1901 ... 2037)
- **%y**: The year without the century (00 ... 99)
- **%m**: Month number (01 ... 12)
- **%d**: Day of the month (01 ... 31)
- **%H**: Hour (00 ... 23)
- **%M**: Minute (00 ... 59)
- **%S**: Seconds (00 ... 59)

When no format string is defined, the following default format is used:

```
%Y%m%d%H%M%S
```

AscDecimal

Use **AscDecimal** to read and write decimal values to and from ASCII streams.

```
AscDecimal( [Integer len [, Bool withPoint [, Integer precision [, Char pointChar [, Identifier
rounding[, Char padChar]]]]]] )
```

- **len**: The total length of the decimal value (default is 0 => unspecified).
- **withPoint**: Boolean flag to specify if there is a decimal point in the string (default is true).
- **precision**: Number of digits after the decimal point (default is 6).
- **pointChar**: Character used as decimal point (default is the point '.').
- **rounding**: Rounding method to use (PLAIN, UP, DOWN, BANK) (default is DOWN).
- **padChar**: Padding character to use (default is '0').

AscInteger

Use **AscInteger** to read and write integer values to and from ASCII streams.

Integer values are supported in the range from -2147483648 to 2147483647.

> **Note:**
>
> AscInteger cannot be NULL (empty).

```
AscInteger( [Integer len [, Char padChar]] )
```

- **len**: The total length of the integer value (default is 0 => unspecified)
- **padChar**: The character used to pad integer values to a fixed length (default is the '0')

AscString

Use the AscString data type to read and write strings to and from an ASCII stream.

```
AscString( [Integer len [, Char padChar [, Bool isLeftJustified]]] )
```

- **len**: The total length of the string (default is 0 => unspecified).
- **padChar**: The character used to pad string values to a fixed length (default is a space character).
- **isLeftJustified**: Flag indicating that the string is left justified (default is **true**).

AscRawString

Equivalent to **AscString**, but preserves leading and trailing spaces while **AscString** strips all spaces from strings.

ASN.1 Data Types

Pipeline Manager supports the following ASN.1 data types:

- ASN_Integer
- ASN_LegacyOctetString
- ASN_OctetString
- ASN_RawOctetString
- ASN_BcdString
- ASN_NumberString
- ASN_HexString
- ASN_Tag
- ASN_Blob

ASN_Integer

Use **ASN_Integer** to read and write integer values from and to ASN.1 streams. Integer values are supported in the range from -2147483648 to 2147483647.

> **Note:**
>
> ASN_Integer cannot be null (empty).

```
ASN_Integer( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application**
  - **Context**
  - **Private**
  - **Universal**

  The default is **Application**.

ASN_LegacyOctetString

Use ASN_LegacyOctetString to read and write an Octet string, which is a Byte string without any specific encoding for the data, for example, ascii or hex, from and to ASN.1 streams. ASN_LegacyOctetString removes the leading and trailing spaces after decoding an octet string.

This data type is similar to the ASN_OctetString data type except for the following difference:

- ASN_LegacyOctetString encodes an empty octet string with length = 0 and no value.
- ASN_OctetString builds an empty octet string with length = 1 and a space for the value. See "ASN_OctetString".

```
ASN_LegacyOctetString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_OctetString

Use ASN_OctetString to read and write strings from and to ASN.1 streams. An Octet string is a Byte string without any specific encoding for the data, for example, ascii or hex.

```
ASN_OctetString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_RawOctetString

Use ASN_RawOctetString to read and write an Octet string, which is a Byte string without any specific encoding for the data, for example, ascii or hex, from and to ASN.1 streams. Unlike the ASN_LegacyOctetString, ASN_RawOctetString does *not* remove the leading and trailing spaces after decoding an octet string.

See also "ASN_LegacyOctetString".

```
ASN_RawOctetString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**

  - **Private**
  - **Universal**

ASN_BcdString

ASN_BcdString is an extension of the **ASN_OctetString** used to read and write strings containing data coded in the Binary Coded Decimal form to and from ASN.1 streams. This type automatically decodes and encodes BCD, so the user accesses the data seamlessly.

```
ASN_BcdString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**

ASN_NumberString

ASN_NumberString is an extension of the **ASN_OctetString** used to read and write strings containing only numbers (and spaces) but packed in an ascii string from ASN.1 streams. An **ASN_NumberString** can be read and written as a string, date, or long.

```
ASN_NumberString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application (Default)**
  - **Context**
  - **Private**
  - **Universal**

ASN_HexString

ASN_HexString is an extension of the **ASN_OctetString** used to read and write strings containing data coded in the Hexadecimal form, but stored as ASCII strings, from and to ASN.1 streams. This type is used because iScript cannot directly manipulate hexadecimal byte strings, so the strings are stored as ASCII representation of hexadecimal strings. For example, 0x28F3 is stored as 28F3.

ASN_HexString supports cases in which a special conversion method of read or write access is necessary.

```
ASN_HexString( Integer TagValue [, String Asn1Class] )
```

- **TagValue**: The value to use as ASN.1 Tag.
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**

  - **Universal**

ASN_Tag

Use **ASN_Tag** to read and write constructed ASN.1 objects to and from ASN.1 streams. Only the Parser should create this type of objects, out of record definitions in the block description file.

The **ASN_Tag** object can read both definite and indefinite length ASN.1 objects.

ASN_Blob

ASN_Blob is a special type used to store a complete structured (constructed) ASN.1 Object in the form of a byte string. This is useful when you need to transmit a block of data from the input to the output without processing, thus not needing to map the data into EDR container fields.

The only limitation for this type is that the ASN.1 object must have a definite length.

```
ASN_Blob( Integer TagValue [, String Asn1Class [, String Asn1Form]] )
```

- **TagValue**: The value to use as ASN.1 Tag
- **Asn1Class**: The Class of the ASN.1 object. Values are:
  - **Application** (Default)
  - **Context**
  - **Private**
  - **Universal**
- **Asn1Form**: The Form of the ASN.1 object. Values are:
  - Constructed (Default)
  - Primitive

TAP Data Types

Pipeline Manager supports the following TAP data types. These are defined only to match the type name used in the TAP format description file.

- **TAP_AsciiString.** Same type as a standard **ASN_OctetString**.
- **TAP_Description.** Same type as a standard **ASN_OctetString**.
- **TAP_Currency.** Same type as a standard **ASN_OctetString**.
- **TAP_PercentageRate.** Same type as a standard **ASN_Integer**.

Setting Up an Input Mapping File

To create an input mapping file, first identify the data in your external files that you need to map from the external files to the EDR container. You can then either start with one of the sample input mapping files or create your own.

Each mapping entry contains a list of mappings either from data record fields to EDR container fields or from constant values to EDR container fields. You can map a data record field to more than one EDR container field by adding more than one mapping. For example:

```
A_NUMBER    -> DETAIL.A_NUMBER;
.
```

```
.
.
A_NUMBER    -> DETAIL.ASS_GSMW_EXT.A_NUMBER_USER;
```

The following grammar defines the syntax of the input mapping file:

```
<input-mapping-file> ::=    <file-format-mapping>*
<constant> ::=              <integer> | <decimal> | <string>
<constant-mapping> ::=      <constant> "->" <edr-field>
<decimal> ::=               "0..9"* "." "0..9"+
<edr-field> ::=             <identifier> ("." <identifier>)+
<field-mapping> ::=         <field-name> "->" <edr-field>
<field-name> ::=            <identifier>
<file-format-mapping> ::=   <file-format> "{" <record-mapping>* "}"
<identifier> ::=            "a..z,A..Z,_" "a..z,A..Z,0..9,_"*
<integer> ::=               "0..9"+
<mapping-entry> ::=         <field-mapping> | <constant-mapping>
<mappings> ::=              <mapping-name> "{" <mapping-entry>* "}"
<record-mapping> ::=        <record-name> "{" <mappings>* "}"
<string> ::=                "\"" "any character"* "\""
```

Setting Up an Input Grammar File

The input grammar contains iScript statements to create an EDR container.

The syntax of the input grammar file is similar to the syntax used in Yacc grammars. This file defines the grammar of the input data that are parsed and of the iScript statements that are run when a certain symbol is found in the input data stream.

Configuring the Input DataDescription Registry Section

You configure a **DataDescription** section in the registry for each pipeline. The **DataDescription** section includes the following entries:

- **StreamFormats**: Specifies the input stream format file.

  See "About the Order of Listing Stream Format Description Files".
- **InputMapping**: Specifies the input mapping description file.
- **OutputMapping**: Specifies the output mapping description file.

> **Note:**
>
> You specify the input grammar description file in the **Input** section.

This sample shows the **DataDescription** section:

```
DataDescription
  {
   Standard
   {
   ModuleName = Standard
   Module
   {
   StreamFormats
   {
     Format1 = ./formatDesc/Formats/Flist/Flist_v01.dsc
```

```
    }
    InputMapping
    {
     Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_InMap.dsc
    }
    OutputMapping
    {
     Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_OutMap.dsc
    }
    }
    }
}
```

> **Note:**
>
> The DataDescription section also includes an entry for the output mapping file. See "Configuring EDR Output Processing".

About the Order of Listing Stream Format Description Files

Pipeline module instances prioritize the order in which formats are considered for parsing in the order that the stream format description files are listed in the **StreamFormats** section of the registry.

Parsing errors can occur in a pipeline if the stream format description files are listed in the incorrect order. For example, the stream format description file that applies to your custom input stream would be listed in the **StreamFormats** section *before* the output stream format description file.

Configuring the Input Section in the Registry

> **Note:**
>
> When you configure the Input section, you configure the Pipeline Input Controller. See "Input Controller".

To configure the **Input** section in the registry, do the following:

- The **UnitsPerTransaction** entry: Use this entry to improve performance.
- The **INP_GenericStream** module: Specify the input grammar file in this section. See "INP_GenericStream".

  When configuring the INP_GenericStream module, configure one of the following modules as a submodule of the INP_GenericStream module:

  - **EXT_InFileManager**: Configure this module if the pipeline receives input from files. This module manages the input, temporary, and done files. See "EXT_InFileManager".
  - EXT_InEasyDB: Configure this module if the pipeline receives input from a database. See "EXT_InEasyDB".
  - If the pipeline receives input from a prepaid network, configure **EXT_InSocketMgrFlist** for input from flist-based networks.

About Getting Pipeline Input from Files

To configure a pipeline to read data from files, use the EXT_InFileManager module.

When you configure the module, you specify the directories, suffixes, and prefixes for the following files:

- **Input files**: CDR files from the mediation system. The prefix and suffix are used by the input module to identify which files to process.

  The input module checks periodically for files in this folder with the specified prefix and/or suffix.
- **Done files**: Created when a transaction is successfully completed.
- **Error files**: Created after a transaction rollback.

To manage file names, you can specify the following:

- The prefix for temporary files. Temporary files are used as input until the transaction is complete.
- Whether to replace or append prefixes and suffixes.
- The time period (in seconds) for which the input directory must be empty before the EVT_INPUT_DIR_EMPTY event is sent.

See "EXT_InFileManager".

About Getting Pipeline Input from a Database

To get pipeline input from a database, use the EXT_InEasyDB module.

To set up a pipeline for database input:

1. Create a job file that consists of SQL statements. It can also include iRule variables, which are defined in a parameter file. The EXT_InEasyDB module uses the commands to create EDRs.
2. Place the job file in a specific directory. When Pipeline Manager starts, the EXT_InEasyDB module finds the directory from the registry and starts the input process according to the commands in the job file.

You can stop and restart the pipeline after a system crash by configuring a restart file.

You can start the pipeline by using the **ReadDatabase** semaphore. When the module receives a start command while in process, the new SQL command is written to a job file.

The returned values of the database input stream contain all fields of each selected row divided by the configurable delimiter.

To configure the pipeline to read data from a database, see "EXT_InEasyDB".

Specifying the Maximum Errors Allowed in an Input File

You can configure the Output Controller to reject an entire input stream that exceeds a maximum percentage of errors. For example, you can specify to reject an input stream if over 20% of the EDRs have a particular error.

You specify the error threshold by using the **MaxErrorRates** entry in the Output section of the registry file.

> **Note:**
>
> You can also configure a pipeline to reject individual EDRs by using the FCT_Reject module. For information, see "About Standard Recycling" and "Recycling EDRs in Pipeline-Only Systems".

When an input stream exceeds the error threshold, the Output Controller:

- Deletes all output streams associated with the input stream. For example, if a pipeline splits an input stream into five output streams, the Output Controller deletes all five output streams.
- Moves the input stream to the error directory. You define the location of the error directory by using the ErrorPath registry entry. For information, see "EXT_InFileManager".

To set an error threshold:

1. Stop Pipeline Manager, if necessary.
2. Open your registry file in a text editor.
3. Edit the Pipeline Output Controller's **MaxErrorRates** registry entries, making sure you:
   - List all error codes that the Pipeline Output Controller should monitor.
   - Specify an error threshold for each error code. The threshold specifies the maximum percentage of EDRs that are allowed to have the particular error.

   For example, to configure the Output Controller to reject a stream if any of the following are true:
   - Over 10% of the EDRs have an INF_EDR_REJECTED error.
   - Over 8% of the EDRs have an ERR_CUST_NOT_FOUND error.
   - Over 20% of the EDRs have an ERR_CHARGED_ZONE_NOT_FOUND error.

```
Output
{
    ...
    MaxErrorRates
    {
        INF_EDR_REJECTED = 10
        ERR_CUST_NOT_FOUND = 8
        ERR_CHARGED_ZONE_NOT_FOUND = 20
    }
    ...
}
```

4. Save and close the registry file.
5. Restart Pipeline Manager.

Reading TAP Files

Pipeline Manager can read the following TAP versions:

- TAP-0301 from the TD57v3.04.00 specification
- TAP-0303 from the TD57v3.07.01 specification
- TAP-0304 from the TD57v3.08.02 specification

- TAP-0309 from the TD57v3.90 specification
- TAP-0310 from the TD57v3.10.01 specification
- TAP-0311 from the TD57v28 specification
- TAP-0312 from the TD57v30.1 specification
- TAP-0312 from the TD57v32.1 specification

You can specify TAP input grammar files when you set up your input modules. The files are located in the *pipeline_home***/formatDesc/Formats/TAP3** directory. *pipeline_home* is the directory where you installed Pipeline Manager.

> **Note:**
>
> Because the TAP 3.10 standard introduced fundamental changes, files produced according to earlier versions of the TAP standard are not compliant with the TAP 3.10 standard. Therefore, the TAP 3.10 grammar files cannot be used to process previous TAP versions.

Note the following implementation details:

- The following records are generated when TAP is processed by Pipeline Manager:
  - 1 Header record (010): 1 EDR
  - 1 Trailer record (090): 1 EDR
  - 1 GPRS record (040 for SGSN, 042 for GGSN or mixed ticket): 1 EDR
  - 1 mobile supplementary service (MSS) record (029): 1 EDR
  - 1 service center usage (SCU) record (050): 1 EDR
  - 1 value added service (VAS) record (060): 1 EDR
  - 1 content transaction (CONT) record (999): 1 EDR
  - 1 location service (LOCN) record (998): 1 EDR
  - 1 mobile originating call (MOC) record (021): *n* EDRs: one EDR for every basic service used.
  - 1 mobile terminating call (MTC) record (031): *n* EDRs: one EDR for every basic service used.
- For each supplementary service, an SS_PACKET is created and attached to the corresponding EDR. For every charge detail of the VAS array, a charge packet is added to the latest generated EDR. The VAS short description is stored in the DETAIL.ASS_CBD.CP.PRODUCTCODE_USED field.
- The Value added service used block is used to build an associated charge breakdown record containing data for rating.
- The CAMEL service information is stored in the ASSOCIATED_CAMEL_EXTENSION ("700") block of the EDR. The associated charge packets are stored on the same ASS_CBD as others but with the PRODUCTCODE_USED field set to CAMEL to identify them.
- Mobile originating and terminating records (MOC and MTC records) are split into multiple records downstream by the ISC_TapSplitting iScript. This iScript generates one EDR for

every basic service in the basic service used array. For more information, see "ISC_TapSplitting".

Note the following restrictions:

- The size of the ASN.1 string is not checked during input.
- Pipeline Manager does not add the MIN to the EDR.
- Pipeline Manager does not add the electronic serial number (ESN) to the EDR.

The TAP output grammar recognizes all record types generated by Pipeline Manager.

About Customizing Mapping of Flist Fields to Rating EDR Container Fields

To process events received from the Connection Manager (CM), a real-time pipeline converts the event data from flist format to rating EDR format for processing by Pipeline Manager.

BRM provides default flist-to-rating-EDR mappings for GSM, GPRS, and SMS events in the *pipeline_home*/**formatDesc/Formats/Realtime/rate_event.xml** file. When the real-time pipeline starts, the INP_Realtime module uses the descriptions in this XML file to construct an in-memory representation of the flist-to-rating-EDR mapping. The pipeline uses the in-memory mapping to convert incoming flists to rating EDR format.

Table 3-8 displays the XML elements are used in the flist-to-rating-EDR mapping:

Table 3-8 XML Elements in flist-to-rating-EDR Mappings

XML element	Description
OpcodeMap	The root node of the XML document. The XML document can have only one OpcodeMap element. This element requires the *opcode* attribute, which is a unique string used to differentiate between OpcodeMap elements in other XML files.
InMap	The flist-to-rating-EDR mappings for the input flist. The XML document can have only one InMap element. The *containerType* attribute specifies the root-level name of the rating EDR container.
FlistField	A field in the input flist. If the *target* attribute is present, the flist field is mapped to the target field in the rating EDR container.
EdrBlock	Indicates when a rating EDR Block should be created.

Sample input flist:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 12035 15
0 PIN_FLD_EVENT        SUBSTRUCT [0] allocated 52, used 52
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/delayed/session/telco/gsm
1373193266068995136 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 12035 0
1     PIN_FLD_START_T      TSTAMP [0] (1081882800) Tue Apr 13 12:00:00 2004
1     PIN_FLD_END_T        TSTAMP [0] (1081883200) Tue Apr 13 12:00:00 2004
1     PIN_FLD_QUANTITY    DECIMAL [0] 400
1     PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 12
2         PIN_FLD_CALLING_FROM    STR [0] "0049100050"
2         PIN_FLD_CALLED_TO       STR [0] "0049100051"
2         PIN_FLD_USAGE_CLASS     STR [0] "NORM"
2         PIN_FLD_TERMINATE_CAUSE   ENUM [0] 0
1     PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 14
2         PIN_FLD_CALLED_NUM_MODIF_MARK   ENUM [0] 0
```

```
                2         PIN_FLD_DIRECTION          ENUM [0] 0
                2         PIN_FLD_CELL_ID            STR [0] "123456"
                2         PIN_FLD_SUB_TRANS_ID       STR [0] "S"
                2         PIN_FLD_DESTINATION_SID    STR [0] ""
                2         PIN_FLD_NUMBER_OF_UNITS    DEC[0] 1.0
```

The default flist-to-rating-EDR mapping in XML for the above flist:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<OpcodeMap xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./OpcodeMapping.xsd"
opcode="PCM_OP_RATE_PIPELINE_EVENT">
   <InMap containerType="DETAIL">
      <!--CONSTANT EDR DETAIL ITEMS-->
      <EdrField name="DETAIL.RECORD_TYPE" value="020" />
      <EdrField name="DETAIL.DISCARDING" value="0" />
      <!--EVENT FLIST ITEMS-->
      <FlistField name="PIN_FLD_EVENT">
         <!--EVENT POID-->
         <FlistField name="PIN_FLD_POID" format="type" target="DETAIL.EVENT_TYPE" alias="EventType"/>
         <!--ACCOUNT_OBJ-->
         <FlistField name="PIN_FLD_ACCOUNT_OBJ" format="short" target="DETAIL.CUST_A.ACCOUNT_PARENT_ID" />
         <!--CHARGING_START-->
         <FlistField name="PIN_FLD_START_T">
            <EdrField name="DETAIL.CHARGING_START_TIMESTAMP" />
            <EdrField name="DETAIL.NE_CHARGING_START_TIMESTAMP" />
         </FlistField>
         <!--CHARGING_END-->
         <FlistField name="PIN_FLD_END_T">
            <EdrField name="DETAIL.CHARGING_END_TIMESTAMP" />
            <EdrField name="DETAIL.NE_CHARGING_END_TIMESTAMP" />
         </FlistField>
         <!--QUANTITY FIELD-->
         <FlistField name="PIN_FLD_QUANTITY" target="DETAIL.DURATION" />
         <!--TELCO FLIST ITEMS-->
         <FlistField name="PIN_FLD_TELCO_INFO">
            <!--A_NUMBER-->
            <FlistField name="PIN_FLD_CALLING_FROM">
               <EDRField name="DETAIL.A_NUMBER"/>
               <EDRField name="DETAIL.ASS_GSMW_EXT.A_NUMBER_USER" onAliasName="EventType" onAliasValue="/
event/delayed/session/telco/gsm" />
            </FlistField>
            <!--B NUMBER-->
            <FlistField name="PIN_FLD_CALLED_TO">
               <EDRField name="DETAIL.B_NUMBER"/>
               <EDRField name="DETAIL.ASS_GSMW_EXT.DIALED_DIGITS" onAliasName="EventType" onAliasValue="/
event/delayed/session/telco/gsm" />
            </FlistField>
            <!--USAGE_CLASS-->
            <FlistField name="PIN_FLD_USAGE_CLASS" target="DETAIL.USAGE_CLASS" />
            <!--CALL_COMPLETION_INDICATOR-->
            <FlistField name="PIN_FLD_TERMINATE_CAUSE" target="DETAIL.CALL_COMPLETION_INDICATOR" />
         </FlistField> <!--END TELCO FLIST ITEMS-->
         <!--GSM FLIST ITEMS-->
         <FlistField name="PIN_FLD_GSM_INFO" onAliasName="EventType" onAliasValue="/event/delayed/
session/telco/gsm" optional="true">
            <EdrBlock name="DETAIL.ASS_GSMW_EXT"/>
            <!--CONSTANT DETAIL.ASS_GSMW_EXT EDR ITEMS-->
            <EdrField name="DETAIL.ASS_GSMW_EXT.RECORD_TYPE" value="520" />
            <FlistField name="PIN_FLD_CALLED_NUM_MODIF_MARK" target="DETAIL.B_M ODIFICATION_INDICATOR" />
            <FlistField name="PIN_FLD_DIRECTION" target="DETAIL.USAGE_DIRECTION" />
            <FlistField name="PIN_FLD_CELL_ID" target="DETAIL.ASS_GSMW_EXT.CELL_ID"/>
```

```
            <FlistField name="PIN_FLD_SUB_TRANS_ID" target="DETAIL.LONG_DURATION_INDICATOR" />
            <FlistField name="PIN_FLD_NUMBER_OF_UNITS" target="DETAIL.NUMBER_OF_UNITS" />
          </FlistField> <!--END GSM FLIST ITEMS-->
          <!--END EVENT FLIST ITEMS-->
        </FlistField>
    </InMap>
</OpcodeMap>
```

You can customize the default GSM, GPRS, and SMS mappings and create custom mappings for other types of events. To customize flist-to-rating-EDR mappings, you must be familiar with the following topics:

- BRM flists.
- XML
- XML Schema

To create and use a custom mapping:

1. Edit the pipeline event XML file (*pipeline_home***/formatDesc/Formats/Realtime/rate_event.xml)** to add your custom mappings.
2. Validate the XML file using *pipeline_home***/formatDesc/Formats/Realtime/opcode_ifw_mapping.xml** file.

> **Note:**
>
> Using an invalid XML file prevents the INP_Realtime module from successfully starting. Make sure you validate the XML file against the XML schema.

About the POID Format in the Rating EDR Container

If an flist field is a POID, the *format* attribute is required to indicate the format of the POID in the rating EDR container. The following format types are supported: long, short, id, and type.

For example, the format of POID 0.0.0.1 /account 12065 is mapped as follows:

- **1_12065 /account** when the format attribute is *long*
- **1_12065** when the format attribute is *short*
- **12065** when the format attribute is *id*
- **/account** when the format attribute is *type*

Mapping an Flist Field to Multiple Rating EDR Container Fields

If the FlistField element contains child **EDRField** elements, the flist field is mapped to multiple rating EDR fields.

In the following example, PIN_FLD_END_T is mapped to multiple fields in the DETAIL EDR block:

```
<FlistField name="PIN_FLD_END_T">
<EdrField name="DETAIL.CHARGING_END_TIMESTAMP" />
<EdrField name="DETAIL.NE_CHARGING_END_TIMESTAMP" />
</FlistField>
```

Using Conditions to Map an Flist Field to a Rating EDR Container Field

You can use the FlistField attributes *alias*, *onAliasName*, and *onAliasValue* for conditional mappings.

In this example, the input flist is a **/event/delayed/session/telco/gsm** event. The PIN_FLD_GSM_INFO substruct of the event is mapped to the DETAIL.ASS_GSMW EDR block.

```
<FlistField name="PIN_FLD_POID" format="type" target="DETAIL.EVENT_TYPE" alias="EventType"/>
<!--GSM FLIST ITEMS-->
<FlistField name="PIN_FLD_GSM_INFO" onAliasName="EventType" onAliasValue="/event/delayed/session/telco/
gsm" optional="true"> <EdrBlock name="DETAIL.ASS_GSMW_EXT"/>
<!--CONSTANT DETAIL.ASS_GSMW_EXT EDR ITEMS-->
<EdrField name="DETAIL.ASS_GSMW_EXT.RECORD_TYPE" value="520" />
<EdrField name="DETAIL.ASS_GSMW_EXT.TIME_BEFORE_ANSWER" value="0" />
<EdrField name="DETAIL.ASS_GSMW_EXT.NUMBER_OF_SS_PACKETS" value="0" />
<FlistField name="PIN_FLD_CALLED_NUM_MODIF_MARK" target="DETAIL.B_MODIFICATION_INDICATOR" />
<FlistField name="PIN_FLD_DIRECTION" target="DETAIL.USAGE_DIRECTION" />
<FlistField name="PIN_FLD_CELL_ID" target="DETAIL.ASS_GSMW_EXT.CELL_ID" />
<FlistField name="PIN_FLD_SUB_TRANS_ID" target="DETAIL.LONG_DURATION_INDICATOR" />
<FlistField name="PIN_FLD_NUMBER_OF_UNITS" target="DETAIL.NUMBER_OF_UNITS" />
</FlistField> <!--END GSM FLIST ITEMS-->
```

You can also map an alias to an **EdrField** defined elsewhere. For example:

```
<FlistField name="PIN_FLD_START_T" target="DETAIL.CHARGING_START" alias="startTime"/>
<!--GSM FLIST ITEMS-->
<FlistField name="PIN_FLD_GSM_INFO" >
<EdrBlock name="DETAIL.ASS_GSMW_EXT"/>
<!--CONSTANT DETAIL.ASS_GSMW_EXT EDR ITEMS-->
<EdrField name="DETAIL.ASS_GSMW_EXT.CHARGING_START" useAlias="startTime" />
</FlistField> <!--END GSM FLIST ITEMS-->
```

4
Configuring EDR Output Processing

This chapter describes how to set up output processing for the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. The input can be CDRs for telco rating.

About the Output Process

Pipeline Manager can generate data for various purposes:

- To provide rated events for the Rated Event (RE) Loader to load into the BRM database.
- To collect rejected EDRs for recycling.
- To collect duplicate EDRs.
- To send data to the Pipeline Manager database for additional processing in another pipeline.
- To handle discarded EDRs.

The output process works as follows:

1. The completed EDRs are moved to the output buffer.
2. The output module reads data from the output buffer and processes the data. The processing performed depends on the output configuration, for example:
   - If the output consists of rated EDRs, the output module uses grammar and description files to convert the EDRs to a format that the RE Loader uses.
   - If the output consists of rejected EDRs, the output module creates reject files.
3. The output module writes the data to the specified output stream. The destination of an output stream can be a file directory, a location in the database, or an external network. You can configure different directories, file names, and network destinations for each output stream.

About the Output Processing System Components

Output processing is managed by the output controller and the output collection module.

- The *output controller* manages the overall output process. You can configure output properties that apply to all streams. See "Output Controller".

  The Output controller performs the following functions:

  – Manages the output stream's trailer information.

  – Rejects input streams when they exceed a specified maximum number of errors.

  – Checks for duplicate CDR files.

  – Notifies the Transaction Manager when a transaction ends.

  – Stops the pipeline when critical errors occur.

  The input stream's trailer information can become invalid when a pipeline discards or rejects individual EDRs or when a pipeline splits EDRs into multiple output streams. For

example, the input stream's trailer contains a total charge amount field, which contains the charge amount for all EDRs. When a pipeline discards some EDRs in the input stream, the total charge amount is no longer valid. To correct this, the Output Controller automatically recalculates the trailer information for each output stream.

- *Output collection* defines all the output streams for the pipeline. The Output Collection module performs the following functions:
  - Generates the output devices that are specified in the registry at system startup.
  - Passes the EDR container to the specified output device.

  You configure the Output Collection module by editing a pipeline's **OutputCollection** section of the registry file. For information, see "Output Controller".

About the Output Modules

The following output modules are available:

- **OUT_GenericStream**: Used to process rated events. This module converts the EDR format to a format needed for further processing, for example, RE Loader format to load rated events or TAP format for outcollect processing of roaming records.

  See "Configuring Output for Rated Events" and "OUT_GenericStream".
- **OUT_Reject**: Used to process rejected or duplicate events. See "Configuring Output for Rejected or Duplicate EDRs" and "OUT_Reject".
- **OUT_DB**: Used to send data to the database. See "Sending Output to a Database" and "OUT_DB".
- **OUT_DevNull**: Used to discard EDRs that you don't want to recycle. See "Configuring Output of Discarded EDRs" and "OUT_DevNull".
- **EXT_OutFileManager**: Used to manage the output files of the OUT_GenericStream module and the OUT_Reject modules. See "EXT_OutFileManager".

About Output Processing File Types

Output processing uses the following types of files:

- **Temporary data file**: Stores records during processing. There is one temporary file for each external file being processed. If the entire external file is rejected, the temporary file is not used.
- **Output File**: Stores the EDRs after processing. This file is derived from the temporary file. When processing is completed successfully, the temporary file is renamed and becomes the output file.
- **Temporary stream file list**: Stores the names of temporary data files. This is required when the Replace registry entry is enabled. In that case, the output file name is appended with a sequence number. Each output stream uses a separate temporary stream file list. For example, the telephony output stream registry includes these entries:

```
TempDataPath       = ./samples/wireless/data/telout
TempDataPrefix     = tel.tmp.
TempDataSuffix     = .data
```

> **Note:**
>
> These registry entries are used for internal pipeline processing only and should not be changed.

These output processing files are managed by the "EXT_OutFileManager" module.

About ASN.1 Output

ASN.1 objects are composed of a triplet TLV (Tag/LengthOfValue/Value). Therefore, before writing ASN.1 to the output, you must calculate the **LengthOfValue** field for every element of the ASN.1 tree that you are building. To do this, use the EXT_AsnTree module.

This extension has functions to perform the following tasks:

- Builds a tree of ASN.1 objects.
- Updates the **LengthOfValue** fields of the ASN.1 objects.
- Flushes the resulting ASN.1 data block to an output stream.

About Configuring Output Processing

To configure output processing, do the following:

1. Create the directories for the output streams.
2. For output of rated events:
   - Set up an output stream format description file.
   - Set up a mapping file.
   - Set up a grammar file.

   In most cases, you need to modify only the sample stream format description and output mapping files.
3. Configure these output sections in the registry:
   - The output mapping in the **DataDescription** section. This specifies the output mapping for rated events. See "Configuring the Output DataDescription Registry Section".
   - The **OutputBuffer** section in the **Pipeline** section.
   - The **Output** section. This section includes the configuration for the pipeline output controller and for output modules such as the OUT_GenericStream module. See "About the Output Modules".
4. Configure the output stream for the function modules that create the output. For example, configure the FCT_Reject module to send rejected EDRs to the reject output stream.

About Configuring the Output Section in the Registry

The output section in a pipeline has this hierarchy:

```
Pipeline
  Output
      Output controller
```

```
Output collection
    Output streams
```

- The *output controller* manages the overall output process. You can configure output properties that apply to all streams. See "Output Controller".
- *Output collection* defines all the output streams for the pipeline. You configure the Output Collection module by editing the **OutputCollection** section of the registry file. For information, see "Output Collection".
- Output streams send EDRs to the appropriate output location. See "About the Output Modules".

About Configuring Statistics Information in the Output Section

> **Note:**
>
> Instrumented statistics are designed for batch rating and not real-time rating. Statistics are related to transactions. Since real-time rating does not use transactions, instrumented statistics are unavailable for real-time rating.

The **Statistic** subgroup controls the statistics related to Pipeline Manager's EDR processing rate. You can view these statistics in the output logs or HTTP browser.

The **Statistic** subgroup has one registry option, **EDRCountCriteria**. This option can have two possible values:

- INPUT: The Pipeline Manager considers only the detail CDRs that are passed through it as input for calculating the EDR statistics.
- ALL: This is the default option. The Pipeline Manager considers all the EDRs that are passed through it for calculating the EDR statistics. This includes the duplicate EDRs created and the EDRs directed to an additional output stream.

> **Note:**
>
> The **Statistic** subgroup is optional. If this subgroup is not present, the behavior of the Pipeline Manager is similar to that when **EDRCountCriteria** is set to ALL.

This sample shows the output hierarchy:

```
Pipelines
{
    W_SAMPLE
    {
        Output
        {
            WriteDefaultEdr = False
            MaxErrorRates
        {
        }
          #The following subgroup is optional
            Statistic
            {
                    EdrCountCriteria = ALL
            }
```

```
            OutputCollection
            {
                Output stream 1
                {
                    ModuleName = OUT_Module_1
                    ...
                }
                Output stream 2
                {
                    ModuleName = OUT_Module_2
                    ...
                }
            } # end of Output
        } # END W_SAMPLE
    } # END Pipelines
```

Configuring Output for Rated Events

To output rated events from pipeline rating, you configure the OUT_GenericStream module. This module converts EDRs to the output format used by RE Loader. The output is a file that is loaded by RE Loader.

To convert data from EDR format to a file, the module uses the following files:

- **Stream format description**
- **Output grammar:** Specifies which records to include in the output.
- **Output mapping**: Specifies how to map the data in EDRs to data in the format defined by the output grammar file.

The sample registry files include stream format description, output grammar, and output mapping files that convert data from the BRM EDR, TAP, and CIBER formats.

To configure the OUT_GenericStream module, you specify the following:

- The output grammar file.
- The BRM event type that the output file contains, such as **/event/delayed/session/gsm**.
- The type of pipeline, for example, rating or backout.
- Whether to delete empty output streams. See "Configuring Pipeline Manager to Delete Empty Output Streams".
- The output module, either "EXT_OutFileManager".

See "OUT_GenericStream".

Creating Separate Output Streams for Each Service

Events from different services (GSM, SMS, and so forth) must be delivered to the RE Loader or a prepaid network in separate files. To do this:

- Configure the IRL_EventTypeSplitting iScript to split EDRs by service code. See "Sending EDRs to Pipeline Output Streams".
- In the registry, configure an instance of the OUT_GenericStream module for each service.

Creating Multiple Output Streams in One Output Registry

Each EDR must have a default output stream and may also contain additional output streams. Use the following iScript functions to manipulate multiple output streams in a single registry:

- edrAddAdditionalStream
- edrRemoveAdditionalStream
- edrGetAdditionalStream
- edrContainsAdditionalStream

To add output streams to the default stream in the sample registry:

1. Create an iScript file that adds the stream.

   The **edrAddAdditionalStream** iScript document contains an example **addoutmod.isc** iScript file that shows how to add two additional output module streams.

2. Reference this iScript file in the pipeline FunctionPool registry section.

   The **edrAddAdditionalStream** iScript document contains an example function registry section that shows how to use the **addoutmod.isc** file to add an output stream in the function registry.

3. Define the stream in the pipeline output registry section.

   The **edrAddAdditionalStream** iScript document contains an example output registry section that shows how to configure additional output streams by using OUT_GenericStream.

See **edrRemoveAdditionalStream** for an example that shows how to remove an EDR output stream.

Configuring the Output DataDescription Registry Section

You configure a **DataDescription** section in the registry for each pipeline. The **DataDescription** section specifies the stream format description and input mapping files (see "About the Input Process") and the output mapping file.

```
DataDescription
  {
   Standard
   {
   ModuleName = Standard
   Module
   {
   StreamFormats
   {
     Format1 = ./formatDesc/Formats/Flist/Flist_v01.dsc
   }
   InputMapping
   {
    Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_InMap.dsc
   }
   OutputMapping
   {
    Mapping1 = ./formatDesc/Formats/Flist/Flist_v01_OutMap.dsc
   }
   }
```

```
    }
}
```

About the Order of Listing Stream Format Description Files

Pipeline module instances prioritize the order in which formats are considered for parsing in the order that the stream format description files are listed in the **StreamFormats** section of the registry.

Parsing errors can occur in a pipeline if the stream format description files are listed in the incorrect order. For example, the stream format description file that applies to your custom input stream would be listed in the **StreamFormats** section *before* the output stream format description file.

Configuring Output for Rejected or Duplicate EDRs

The OUT_Reject module writes rejected EDRs to a reject file. A rejected EDR is identical to the EDR input.

To configure output for rejected EDRs, do the following:

- For rejected EDRs, configure the FCT_Reject module.

  See:

  - Configuring Standard Recycling
  - Recycling EDRs in Pipeline-Only Systems
- For duplicate EDRs, configure the FCT_DuplicateCheck module. See "Handling Duplicate EDRs".
- In the registry, configure an instance of the OUT_Reject module for rejected EDRs and an instance for duplicate EDRs. See "OUT_Reject".

  You can use the same output directory with different file suffixes and/or prefixes for rejected or duplicate EDRs.

The OUT_Reject module is a submodule of the Output Collection module. All registry parameters and error messages are handled by the Output Collection module. See "Output Collection".

File handling for rejected and duplicate EDRs is performed by the EXT_OutFileManager module. See "Sending Output to a File".

Sending Output to a File

To send output to a file, use the "EXT_OutFileManager" module. The EXT_OutFileManager module handles file prefixes, suffixes, and paths for the OUT_GenericStream and OUT_Reject modules.

The EXT_OutFileManager module writes the output data to a temporary data file. When the TAM reports a successful completion of a transaction, the file is renamed to an output file and the temporary data file is removed.

If a transaction is rolled back, the original input is restored, and the temporary data file is moved to an error directory and renamed with an error prefix and/or suffix.

By default, the EXT_OutFileManager module is configured to delete empty output files. There is a short period of time during which empty output files are renamed and deleted. If another

process, such as one that manages output files, tries to manipulate the file during this period, the pipeline may experience errors. To avoid this problem, configure any output file management processes to wait approximately one minute before attempting to manipulate files.

Configuring the Temporary File Name

Use the **TempPrefix** registry entry to specify the prefix for temporary data files.

> **Note:**
>
> Do not change the **TempDataPrefix**, **TempDataSuffix**, and **TempDataPath** registry entries. These entries are used by the pipeline for internal data processing only.

See "EXT_OutFileManager".

Configuring File Prefixes and Suffixes

Use the **OutputPath**, **OutputPrefix**, and **OutputSuffix** registry entries to manage the output files for each stream.

> **Note:**
>
> To ensure output file integrity, specify a unique combination of **OutputPath**, **OutputSuffix**, and **OutputPrefix** values for each output stream defined in the registry.

See "EXT_OutFileManager".

Creating an Output File Name from the Input File Name

To indicate that the input file name will be used to create the output file name, use the **UseInputStreamName** registry item.

For example, when **UseInputStreamName = [2,4;4,6;8,&]**, the following characters from the input file name are used to build the output file name:

- Characters 2 to 4.
- Characters 4 to 6.
- Characters 8 to end of string, where the '&' symbol indicates the end of the string.

For instance, if the name of the input file is **test12345678**, **Outputprefix** is **test**, and **Outputsuffix** is **.edr**, the output file name will be **testestt1245678.edr**

See "EXT_OutFileManager".

Applying a Prefix to the Sequence Number

Use the **SequencerPrefix** registry entry to specify a prefix to the sequence number before it gets appended to the generated output file name.

> **Note:**
>
> This entry is used only when **AppendSequencerNumber** is set to **True**.

For example, when **SequencerPrefix** is "+", **Outputprefix** is test, **Outputsuffix** is **.edr**, the output file name will be **test+000002.edr**.

By default, the **SequencerPrefix** is "_" and if no **SequencerPrefix** is needed, it has to be specified as **SequencerPrefix** = ""

> **Note:**
>
> Do not use the characters #, $, =, /, or \ to specify SequencerPrefix.

Using the Output of One Pipeline as the Input to Another Pipeline

If you want to use the output of one pipeline as the input for another pipeline, you must move the output file to another directory before it can be used by the next pipeline. You move the files by using "Event Handler" and an external script. Event Handler would run your external script when prompted by a specified internal event. The external script would move the output file from one directory to another.

For example, if you want to use the output file from pipeline A as the input file to pipeline B:

- Configure pipeline A to generate output files in directory 1.
- Configure pipeline B to process input files from directory 2.
- Create a simple script (*ExternalScript*) that moves the output files from directory 1 to directory 2.
- Configure Event Handler to run *ExternalScript* whenever pipeline A generates an output file. For example, you can configure Event Handler to run *ExternalScript* when it receives an EVT_OUTPUT_FILE_READY event from the "EXT_OutFileManager" module.

Sending Output to a Database

Use the OUT_DB module to send data to a database. You configure the following:

- A SQL command parameter file
- Files that define the following:
  - SqlBeginStream statement
  - HEADER records
  - DETAIL records
  - TRAILER records
  - SqlEndStream statement
- The OUT_DB module

The OUT_DB module reads the parameter file for each new output stream.

When you configure the OUT_DB module, you configure the following:

- The database to load data into.
- Path and file names for the configuration files.
- Aliases for the stream name and row number.
- Source and destination for the header record.
- File management options.

For more information about the OUT_DB module registry entries, see "OUT_DB".

About the OUT_DB Module Configuration Files

The parameter file contains key and value pairs. The keys are used by other configuration files (**SqlBeginStream**, **HeaderTableDefinition**, **DetailTableDefinition**, **TrailerTableDefinition**, **SqlEndStream**). The syntax to recognize keys is ${KEY}. When the other files are processed, all found keys are replaced by the corresponding values. The files are not deleted.

It is possible to change the database tables in a running system if there are no open streams.

The **SqlBeginStream** file and the **SqlEndStream** file are used to define SQL statements that are passed to the database. The begin stream statement is run before the first EDR arrives, and the end stream statement is run after the last EDR is processed.

The **HeaderTableDefinition**, **DetailTableDefinition** (only if the **NumberOfRows** registry entry is set to **1**), and **TrailerTableDefinition** files are used to describe the table in which the EDRs must be stored. For more information, see "HEADER, TRAILER, and DETAIL Table Definitions".

First the table name is defined with ${TABLE}, which is replaced by the definition in the parameter file. Then the column names of the table are defined. Each line holds a column of the table and its EDR container field, which is mapped into this column. The value is the EDR container field with external and alias names delimited by a comma (**,**).

```
${HEADER_TABLE}
;RECORD_TYPE      = HEADER.RECORD_TYPE   , HDR_RECORD_TYPE
```

Each column starts with the **FieldDelimiter** after the table name.

Because schema definition is not supported by the **RWDBBulkInserter**, when the **NumberOfRows** registry entry is set to **1**, the columns are defined like the example above. For bulk insertion into the database when **NumberOfRows** is greater than 1, the column definition is deleted from the table definition file and the schema of the table definition in the database is used to create the **BulkInsert**. The following example shows the DETAIL table definition file for **BulkInsert**:

```
${DETAILTABLE}
;DETAIL.RECORD_TYPE   , RECORD_TYPE
;DETAIL.RECORD_NUMBER , RECORD_NUMBER
...
```

In this case, each sequence of the EDR container field definition in the table definition file must be in the same order as the column definition in the database table.

Each logical block must end with two number signs (##) except within the table definition file. A comment line must start with two slashes (//).

All these files result in one final configuration file for each stream with replaced parameter keys within the SQL statements, except optional registry keys. The registry keys

(**StreamNameAlias** and **RowNumAlias**) are replaced immediately before the SQL statement is passed to the database.

The name of the final configuration file is the stream name. It is located in the ControlPath. If the **SaveConfigurationFile** registry entry is enabled, a suffix is added. If processing is successful, the suffix is **.done**; otherwise, it is **.err**.

Specifying the Destination

The destination value can be applied to an appropriate field in the HEADER record of an output module. In the BRM format, this entry is used to fill the recipient field in the HEADER record.

Specifying the Source

The source value can be applied to an appropriate field in the HEADER record of an output module. In the BRM format, this entry is used to fill the sender field in the HEADER record.

Handling Empty Output Streams

Because of splitting and rejecting, there is a possibility that an output stream may contain only the HEADER and TRAILER records. In this case, the production of a default DETAIL record can be forced. If the **WriteDefaultEdr** registry entry is set to **True**, every DETAIL table contains at least one DETAIL record.

If the **DeleteWithoutDetails** registry entry is set to **True**, all insert and update operations will be rolled back and the default record will be deleted.

These settings are ignored if the output is for a reject stream.

Parameter File

Items in bold text are parameter keys.

```
DETAIL_TABLE
sol42_out
##
HEADER_TABLE
sol42_out_header
##
TRAILER_TABLE
sol42_out_trailer
##
```

HEADER, TRAILER, and DETAIL Table Definitions

HeaderTableDefinition

The format of **HeaderTableDefinition** does not depend on the **NumberOfRows** registry entry. The format of **HeaderTableDefinition** is as follows:

```
${HEADER_TABLE}
;RECORD_TYPE      = HEADER.RECORD_TYPE   , HDR_RECORD_TYPE
```

TrailerTableDefinition

The format of **TrailerTableDefinition** does not depend on the **NumberOfRows** registry entry. The format of **TrailerTableDefinition** is as follows:

```
${TRAILER_TABLE}
;RECORD_TYPE      = TRAILER.RECORD_TYPE   , TRR_RECORD_TYPE
```

DetailTableDefinition

The format of **DetailTableDefinition** depends on the **NumberOfRows** registry entry. When NumberOfRows is set to **1**, the format of **DetailTableDefinition** is as follows:

```
${DETAIL_TABLE}
;record_type      = DETAIL.RECORD_TYPE   , RECORD_TYPE
;record_number    = DETAIL.RECORD_NUMBER , RECORD_NUMBER
```

When NumberOfRows is greater than **1**, the format of **DetailTableDefinition** is as follows:

```
${DETAIL_TABLE}
;DETAIL.RECORD_TYPE   , RECORD_TYPE
;DETAIL.RECORD_NUMBER , RECORD_NUMBER
```

SqlBeginStream

Identifiers in bold text are registry entries. They are replaced before the statement is passed to the database.

One statement:

```
insert into stream_process (streamname, startdate,runmode,numberofrow,destination,source,info)
values
// this is a comment line
(__StreamName__, sysdate,'Debug',500,'${TABLE}','sol42_detailin','testing')
```

More statements:

```
begin
insert into tab1 (col1, col2) values (val1, val2);
insert into tab2 (col1, col2, col3) values
(val1, val2, val3);
// if an SQL block is used there must be a semicolon at the end
end;
```

SqlEndStream

Identifiers in bold text are registry entries. They are replaced before the statement is passed to the database.

```
update stream_process set enddate = sysdate,
edrnum = __RowNum__,
// duration only valid, if startdate and sysdate within one day
duration=
(3600*to_char(sysdate,'HH24')+60*to_char(sysdate,'MI')+to_char(sysdate,'SS'))-
(3600*to_char(startdate,'HH24')+60*to_char(startdate,'MI')+to_char(startdate,'SS'))
where
streamname = __StreamName__
```

Generated Configuration File

```
SqlBeginStream
insert into stream_process (streamname, startdate,runmode,numberofrow,destination,source,info)
values
(__StreamName__, sysdate,'Debug',500,'sol42_out','sol42_detailin','testing')
##
HeaderTableDefinition
```

```
sol42_out_header
##
DetailTableDefinition
sol42_out
##
TrailerTableDefinition
sol42_out_trailer
##
SqlEndStream
update stream_process set enddate = sysdate,
edrnum = __RowNum__,
duration=
(3600*to_char(sysdate,'HH24')+60*to_char(sysdate,'MI')+to_char(sysdate,'SS'))-
(3600*to_char(startdate,'HH24')+60*to_char(startdate,'MI')+to_char(startdate,'SS'))
where
streamname = __StreamName__
##
```

5

Configuring EDR Preprocessing

This chapter describes how to configure the modules used for preprocessing event data records (EDRs) in the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager; for example, it describes how to discard duplicate EDRs.

Handling Duplicate EDRs

Use the FCT_DuplicateCheck module to find EDRs that have already been processed and send them to a special output stream. This prevents you from charging a customer twice for the same usage.

As a pipeline processes EDRs, the following occurs:

1. The FCT_DuplicateCheck module keeps a record of EDRs that have already been processed.
2. As new EDRs are processed, the module checks for duplicate EDRs by comparing data in the incoming EDRs with data in the EDRs that have already been processed.

   The FCT_DuplicateCheck module uses the following criteria to check for duplicate EDRs:

   - **The date of an EDR.** If an EDR is older than a certain date, it is not processed and no further checking is performed. It is unlikely that a duplicate EDR will be processed much later than an original EDR.
   - **The data in an EDR.** You configure which data to use when comparing a new EDR against EDRs that have already been processed.
   - **A search key**. An EDR is considered a duplicate if it has the same search key and the same values contained in the fields used for data comparison.
3. If an EDR is a duplicate, it is flagged with an error so other modules do not need to process it, and it is moved to a separate output stream.

Configuring Duplicate EDR Checking

To enable duplicate EDR checking, configure the FCT_DuplicateCheck module. See "FCT_DuplicateCheck" and the following topics:

- Setting Date Parameters for Storing Processed EDRs
- Specifying the Fields to Use for Duplicate Check
- Specifying a Search Key for Duplicate Check
- Managing FCT_DuplicateCheck Data Files
- About Storing EDRs in a Database Instead of Files
- Using Duplicate Check with Multiple Pipelines
- Suspending Duplicate EDRs

To define the output stream for duplicate EDRs, use the **Output** configuration in the registry. You typically use the OUT_Reject module. See "Sample Output Configuration" and "OUT_Reject" for more information.

> **Note:**
>
> When you enable duplicate checking, you must also set the transaction manager **RedoEnabled** entry to **True**.

Setting Date Parameters for Storing Processed EDRs

To check for duplicate EDRs, you need to store a record of the EDRs that have already been processed. The FCT_DuplicateCheck module then checks incoming EDRs against the previously processed EDRs. If you receive a high volume of EDRs, you cannot store a record of *all* EDRs. Therefore, to limit the number of EDRs you store, you specify date criteria. If an EDR is older than the specified date, it is not processed.

> **Note:**
>
> The date of an EDR is derived from its DETAIL.CHARGING_START_TIMESTAMP field.

To specify date parameters for EDR storage, you use the following date settings, specified in the FCT_DuplicateCheck registry. Depending on an EDR's date relative to these settings and on whether the FCT_DuplicateCheck module is connected to the database, the EDR is ignored, stored in a file, stored in the database, or stored in memory.

- **StoreLimit**: Specifies the oldest date that previously-processed EDRs can be stored. EDRs dated earlier than the **StoreLimit** date are ignored and not processed by the FCT_DuplicateCheck module.
- **BufferLimit**: Specifies the oldest date that previously-processed EDRs can be stored in memory. All EDRs whose date is equal to or later than the **BufferLimit** date are stored in memory. The FCT_DuplicateCheck module searches the memory directly, thus improving performance.

The **StoreLimit** date must be equal to or earlier than the **BufferLimit** date. For example, if the **StoreLimit** is June 1, the **BufferLimit** date can be June 1 or later.

> **Note:**
>
> Because **StoreLimit** and **BufferLimit** are specified as absolute dates (for example, August 2, 2004), you need to change them daily. You change them by using an FCT_DuplicateCheck semaphore file entry. See "FCT_DuplicateCheck" for more information.

Specifying the Fields to Use for Duplicate Check

You configure which data to use when comparing a new EDR against EDRs that have already been processed. A typical duplicate check for a phone call compares the A number, B number, and start time in a new and processed EDR. If the data in all fields match, the new EDR is flagged as a duplicate.

To specify which fields to use, use the FCT_DuplicateCheck **Fields** registry entry. See "FCT_DuplicateCheck" for more information.

The following example shows a typical configuration:

```
Fields
{
    1   =   DETAIL.BASIC_SERVICE
    2   =   DETAIL.B_NUMBER
}
```

> **Note:**
>
> Do not use the DETAIL.CHARGING_START_TIMESTAMP field for duplicate checking.

Specifying a Search Key for Duplicate Check

The duplicate check search key identifies duplicate EDRs. An EDR is considered a duplicate if it has the same time, the same search key value, and the same values for fields listed in the IFW_DUPLICATECHECK table or in the **Fields** lists stored as a record in memory.

The search key is used as a key to the internal data structure. For example, if the search key is **A_NUMBER**, then **A_NUMBER** is the hash key used to find the data in memory or in a file for the EDR being checked.

Managing FCT_DuplicateCheck Data Files

The FCT_DuplicateCheck module uses data files to store EDR data while the EDRs are processed. The file name syntax is:

File_Name_Transaction Id **.dat**

File_Name is the value entered in the **FileName** registry entry:

```
FileName = duplicateData
```

The file contains the data from the fields specified in the registry and the EDR date. At the end of each transaction, the FCT_DuplicateCheck module saves the data from memory to disk in a new transaction file.

> **Note:**
>
> - To store EDRs in files for duplicate checking, configure unique **FileName** settings, **Path** registry settings, or both for each module.
> - To store EDRs in the database for duplicate checking, use the FCT_DuplicateChecking module's **DataConnection** registry entry to connect the module to the database. See "About Storing EDRs in a Database Instead of Files".

The duplicate check transaction files should be backed up routinely when the pipeline is not processing EDRs. Temporary files, however, should not be backed up.

To restore transaction files from a backup, shut down the pipeline and restart after restoring backed up transactions files. You can restore duplicate check data without shutting down provided that the pipeline is not processing any EDRs. There is no semaphore to reload the data file, but resetting the **StoreLimit** or **BufferLimit** settings results in a reload. The values of these settings do not need to be changed from their original startup registry settings.

After an abnormal termination, temporary files may be left behind (there should be only one because file mode should only be used with a single pipeline). These files correspond to transactions that were never committed, and the input files associated with these transactions will be reprocessed upon restarting. You should delete these temporary files before restarting. Temporary files use the suffix **.tmp**.

About Storing EDRs in a Database Instead of Files

If you use the FCT_DuplicateCheck module's **DataConnection** registry entry to connect the module to the database, the module handles EDRs as follows:

- EDRs whose date is equal to or later than the **StoreLimit** date and earlier than the **BufferLimit** date are stored in the database instead of in files. If an EDR is stored in the database, files are not created.
- EDRs whose date is equal to or later than the **BufferLimit** date are stored in memory and in files.

> **Note:**
>
> To avoid using excessive disk space when checking for duplicate EDRs, use the FCT_DuplicateCheck module's **DataConnection** registry entry to connect the module to the database. (The **StoreLimit** and **BufferLimit** date settings, and connecting the module to the database, are designed to help maintain performance without overloading memory.)

When the FCT_DuplicateCheck module is connected to the database, an entry is added to the IFW_DUPLICATECHECK database table for each unique EDR. For each duplicate EDR, the error INF_DUPLICATE_EDR is reported and no entry is added to the table.

If the pipeline transaction is canceled, all the rows with the current transaction ID are removed.

If you do not use the FCT_DuplicateCheck module's **DataConnection** registry entry to connect the module to the database (the default), the module handles EDRs as follows:

- EDRs whose date is equal to or later than the **StoreLimit** date and earlier than the **BufferLimit** date are stored in files instead of the database. (Each EDR is assigned a value that is stored in memory. If the EDR is stored in files, the module checks the memory for the value, which points to the EDR in the file. The module then reads the EDR data from the file.)
- EDRs whose date is equal to or later than the **BufferLimit** date are stored in memory and in files.

> **Note:**
>
> Although not connecting the module to the database enables faster checking for duplicate EDRs, it uses a large amount of disk space.

Create Database Tables for Duplicate Check Data

Use the FCT_DuplicateCheck **TableSuffix** registry entry to create multiple IFW_DUPLICATECHECK tables when you run multiple pipelines. You typically do this when you run a separate pipeline for each type of service.

For example, if the pipeline processes GSM EDRs, you can use GSM as the table suffix to use a table named IFW_DUPLICATECHECK_GSM.

You need to manually create the tables and the indexes. For example:

- IFW_DUPLICATECHECK_GSM
- BIDX_DUPCHK_DATA_GSM

Using Duplicate Check with Multiple Pipelines

When you use the FCT_DuplicateCheck module, you must process the EDRs for each account in the same pipeline. If you use duplicate check in multiple pipelines for the same account, duplicate calls may get processed by different pipelines and will not be recognized as being duplicate.

Suspending Duplicate EDRs

By design, a duplicate EDR is not considered as a suspended EDR and is sent to a different output stream than suspense handling stream. Therefore, duplicate EDRs are not handled by suspense handling.

If your business requires to process duplicate EDRs as suspended EDRs, you can do the following:

1. Change the entry **StreamName = DuplicateOutput** to **StreamName = SuspenseCreateOutput** in the FCT_DuplicateCheck registry.

   This will cause the duplicate check reject file to be routed to the suspense handling stream.

2. Add an iScript in the pipeline after duplicate check processing, to check for EDR error. If the error is duplicate check error, set the error to some other error and set error status as major.

   This will cause the duplicate EDR to be processed by FCT_Reject and FCT_Suspense plugins.

> **Note:**
>
> You should ensure that duplicate EDRs are associated with a specific error code to be able to manage and monitor the EDRs in Suspense Manager Center.

Assembling EDRs

Use the FCT_CallAssembling module to assemble multiple CDRs into a single EDR that pipeline modules can rate. You typically need to assemble CDRs for long phone calls or GPRS sessions that have been recorded in multiple CDRs.

The default behavior of FCT_CallAssembling is designed to assemble *time duration* calls. This is appropriate for wireless voice calls that are rated based on how long the call lasts. You can also configure this module to assemble calls in a manner appropriate for EDRs rated based on

the *volume of data sent*. This is appropriate for a long data transfer session, such as downloading a movie. You can choose to collect both time duration and data volume from multiple CDRs, and a number of other matrixes, such as:

- Volume sent
- Volume received
- Number of units
- Retail charge amount
- Wholesale charge amount

For example, a GPRS session might last for 24 hours. The network might be configured to generate an intermediate CDR every 30 minutes. This GPRS data session is recorded by several partial CDRs. If you rate by volume per session, you use the FCT_CallAssembling module to assemble the partial CDRs into one EDR before rating.

However, remember that the more metrics that you collect, the more system resources you will use.

How FCT_CallAssembling Classifies EDRs

FCT_CallAssembling uses the LONG_DURATION_INDICATOR EDR field to classify assembled calls.

The purpose of LONG_DURATION_INDICATOR changes when calls are assembled. CDRs arrive with this field set to one of these values, which identify the type of call segment:

- **F**: The first segment of the call.
- **L**: The last segment of the call.
- **I**: An intermediate segment of the call.

After FCT_CallAssembling assembles these call segments into an EDR, it gives the EDR one of these long duration indicators to specify its status.

- **C**: Complete call. The first and last segments have both arrived, enabling the call's duration to be calculated. If intermediate call segments arrive after the call is complete, they are given a long duration indicator of **XC**, **XO**, or **XP** and may not be rated.
- **S:** A single CDR containing the entire call.
- **SL**: Slice of a call. Used if **KeepCallOpen = True** or **MaxDuration = True**. Composed of the first call segment and any intermediate call segments that have arrived when the call is flushed. This part of the call is rated. When any other intermediate segments and the last segment arrive, the call is given a long duration indicator of **C** (complete), and these segments are rated.
- **P**: Partially assembled call. Used if **FlushLimit = True** or **KeepCallOpen = false**. Partially assembled calls are rated with the information in whatever segments have arrived. Subsequent call segments are given a long duration indicator of **XP** and are not rated.
- **XC:** Late intermediate EDR. This long duration indicator is for intermediate call segments that arrive after the call is marked complete and rated. Late intermediate call segments are not rated; instead they are used for auditing.
- **XO**: Late overlap EDR. Used if time duration rating is used, and **DropLateCDRs=False**. This status indicates that the call segment was flushed before this CDR arrived, and it represents a time duration period already rated. These late segments are not rated; instead they are used for auditing.

- **XP**: Late timeout EDR. Used if **DropLateCDRs** = **False**. This status indicates that the call timed out before this CDR arrived. These late segments are not rated; instead they are used for auditing.

Managing the Call Assembling Data Files

Open EDRs are stored in a data file. You configure the path and file name in the **Path** and **FileName** entries in the FCT_CallAssembling registry. The syntax for the file name is:

*File_Name_Transaction_ID***.dat**

This file is read at startup and reloaded after rollback. While processing, data is stored in a temporary file.

The FCT_CallAssembling module processes the EDRs in a single transaction. You should backup the work files routinely when the pipeline is not processing any EDRs.

To restore from a backup, shut down the pipeline and restart after restoring backed up work files. Work files use the suffix **.dat**.

After abnormal termination, temporary files may be left behind. Temporary work files use the suffix **.tmp**. These correspond to transactions that never committed. Therefore, the input files associated with these transactions will be reprocessed upon restarting. Delete these temporary files before restarting.

> **Note:**
>
> Never delete the most recent **.dat** file.

Configuring Call Assembling

To configure call assembling, see "FCT_CallAssembling" and the following topics:

- Rating Calls by Time Duration
- Rating Calls by Implied Time Duration
- Rating Calls by Volume of Data Sent

In addition to specifying how to perform call assembly, you can configure FCT_CallAssembling to do the following:

- Specify an amount of time (in seconds) that is an acceptable amount of time error for each call. See "Specifying a Time Error" for more information.
- Keep calls open indefinitely, and rate them in segments periodically. See "Rating Continuous Data Calls by Segment" for more information.
- Limit the effect of **FlushLimit** to calls with specific service codes. See "Rating Partial Calls by Service" for more information.
- Capture data from **Basic Detail Record** fields for the **L** call segment. See "Capturing Fields From the Last Call Record " for more information.
- Get a report about calls being assembled. See "Tracking the Status of Assembled Calls" for more information.

If you are upgrading, see "Migrating Call Assembling Data Between Releases and Pipelines".

Rating Calls by Time Duration

By default, the FCT_CallAssembling module assembles the time duration of a call so it can be rated.

To assemble EDRs for their time duration, the FCT_CallAssembling module identifies partial EDRs by using the following EDR attributes:

- **The chain reference.** This ID identifies which event the partial EDR belongs to. Multiple partial EDRs that belong to the same event all have the same chain reference.

  > **Note:**
  >
  > Chain reference must be unique for each call instance. Oracle does not support identical chain references across several call events.

- **The long duration indicator.** For a list of these indicators, see "How FCT_CallAssembling Classifies EDRs".

Parts of EDRs can be processed out of order; for example, the Last segment might arrive before the First segment. The FCT_CallAssembling module manages EDRs by tracking their status:

- As a soon as a first or intermediate call segment record arrives, the EDR is stored in a data file and the state is set to *Open*. See "Managing the Call Assembling Data Files".
- By default, if a First or Last segment is already stored in a file, and the matching Last segment or First segments arrive, the record state is changed to *Closed*. The EDR is moved back into the pipeline. Any Intermediate segment that belong to the assembled EDR that arrive after the assembled EDR is closed are ignored.

  You can also direct this module to wait for all CDRs before closing a call, or close it after you send in a semaphore. This allows you to rate incomplete calls or calls that never receive a first or last record.

The time duration for a call is calculated as follows:

```
Call Duration = [(Start_Time_Of_Last_Segment) - (Start_Time_Of_First_Segment)] +
Duration of the Last Segment
```

For example, if there is a call which starts at Jan 01, 2009 12:00:00 PM and ends at Jan 01, 2009 at 12:30:00 PM. The EDR will have the following information:

- First Segments TimeStamp as Jan 01, 2009 12:00:00 PM and Duration
- Last Segment TimeStamp would be Jan 01, 2009 12:25:00 PM + 300(Seconds)

In this case, the call duration will be calculated as follows:

```
[(12:25:00 PM) - (12:00:00 PM)] + 300 = 1800 Seconds
```

Rating Incomplete Time Duration Calls

Some EDRs can never be completely assembled because the First, Intermediate, or Last segment never arrives. In this situation, you can choose to close these calls using either the **FlushLimit** semaphore entry or the **MaxDuration** startup registry entry. Both of these entries force calls to be flushed and rated after the time limit you set.

Here is a comparison of the two entries:

- **MaxDuration** is a *startup* registry entry. It takes effect when you start the pipeline. Each time a new call segment arrives, it recalculates the total time duration for that call. If the new time duration exceeds the limit you set, the call is flushed and it remains open for more CDRs. You set the time duration in seconds. To change the setting, you must restart the pipeline.

  For details, see "Using MaxDuration to Rate Incomplete Calls".

- **FlushLimit** is a *semaphore* registry entry used with **KeepCallOpen=True**. It sets a maximum age a call can have before being flushed. When you send in the semaphore, the pipeline calculates whether a call exceeds the maximum age. If a call exceeds the **FlushLimit** setting, FCT_CallAssembling creates and rates the EDR and then closes the call. You set the time limit in days. A setting of **0** flushes all calls. A setting of **1** flushes all calls that have been opened more than 1 day. You can change the **FlushLimit** setting every time you send in a semaphore.

  For details, see "Using FlushLimit to Rate Incomplete Calls".

Using MaxDuration to Rate Incomplete Calls

MaxDuration is a startup registry entry that directs FCT_CallAssembling to rate segments of a wireless call periodically. This entry specifies the total amount of time duration (in seconds) that a call can have before the call segments that have arrived are rated. FCT_CallAssembling recalculates the call duration for every call each time a new call segment arrives and compares it to the **MaxDuration** setting. If the new time duration equals or exceeds the setting for **MaxDuration,** FCT_CallAssembling creates an EDR to rate the existing portion of the call.

To use the **MaxDuration** entry, add it to the startup registry and start (or restart) the pipeline.

Table 5-1 illustrates FCT_CallAssembling behavior with **MaxDuration** set. In the example, the CDRs did not arrive in chronological order.

Example call with KeepCallOpen=True and MaxDuration=28800 (8 hours)

Table 5-1 FCT_CallAssembling and CDRs

CDR	Start time	End time	Duration (hours)	FCT_CallAssembling takes this action...
F	1:00	4:00	3	Waits. **MaxDuration** setting not reached.
I1	4:00	7:00	3	Waits. **MaxDuration** setting not reached.
I2	7:00	10:00	3	Creates EDR for 9 hours of call duration. **MaxDuration** setting exceeded.
I3	10:00	13:00	3	Waits. **MaxDuration** setting not reached.
I5	16:00	19:00	3	Creates EDR for 9 hours of call duration, even though a CDR is missing. **MaxDuration** setting exceeded.
I4	13:00	16:00	3	Not rated; call duration already rated. This EDR is either emitted as **XO** or dropped, depending on the **DropLateCDRs** setting.

In this example, if **MaxDuration** is set to **28800** seconds (8 hours), FCT_CallAssembling rates the EDRs of a call with a total time duration of more than 8 hours. This call arrives in 3-hour segments and will be rated after the third segment arrives, and FCT_CallAssembling calculates that the 9-hour call duration exceeds the 8-hour limit. When the **MaxDuration**

setting is reached, the call segments are flushed and rated, the long duration indicator for the call is set to **SL**, and the call is left open for more call segments.

If a CDR is missing, FCT_CallAssembling adds the missing call time represented by the EDR if it can. In the example above, the I4 call segment arrived last, but the time duration it represented had already been rated when the I5 segment arrived. FCT_CallAssembling calculated the time duration by subtracting the start time from I3 and the end time from I5. The difference was 9 hours of time duration, which exceeds the 8-hour setting, so an EDR was created to rate that duration.

Using FlushLimit to Rate Incomplete Calls

FlushLimit assembles the calls and emits them into the pipeline for rating. The call has a long duration indicator set to **P** for "Partial."

In this example, **FlushLimit** flushes all calls the have a CHARGING_START_TIMESTAMP older than five days from today.

```
FlushLimit=5
```

To flush all incomplete calls, use 0; for example:

```
FlushLimit=0
```

The flush operation does not happen immediately at the time of semaphore execution. Instead, it happens at the arrival of the next pipeline transaction. This is done to ensure that Flush operations happen within the context of a transaction.

Removing Incomplete Time Duration Calls

When you flush EDRs, they re-enter the pipeline as part of the current transaction. The EDRs are still stored in the work files (**.dat** and **.EDR**), with a state of Timeout. This prevents a late-arriving call segment from re-opening a call that has already been flushed. If you are sure that no further segments will arrive, use the **RemoveLimit** semaphore entry to remove calls from the work files.

The **RemoveLimit** entry removes all calls with the Closed or Timeout status, but leaves calls with a state of Closed_Rejected or Timeout_Rejected alone. Closed_Rejected or Timeout_Rejected calls will be recycled. However, if you are sure that calls with the Closed_Rejected or Timeout_Rejected status will not be recycled, use a **RemoveRejectedLimit** semaphore entry to remove these calls from the work files.

Dropping Late Calls

You can drop late EDRs from the pipeline entirely, or send them through as non-valid EDRs. If you send them through the pipeline, you can use them for auditing.

Use the **DropLateCDR** registry entry to specify how to handle the output of late EDRs:

- **True** (Default) = Drop late EDRs from the pipeline. The EDRs are counted in the report of late-arriving EDRs.
- **False** = Send late EDRs through the pipeline as non-valid. The EDRs are not counted in the report of late-arriving EDRs. They are not rated.

Rating Calls by Implied Time Duration

By default, FCT_CallAssembling calculates a call's time duration by using the difference between the first EDR's start time and the last EDR's end time. This includes the time duration for any CDRs between the first and last that have not yet arrived. The time duration for these missing CDRs is included in the EDR and rated.

If **KeepCallOpen=True** calls are rated in segments. When a semaphore with **KeepCallOpen=True** is sent in:

- All the CDRs that have arrived for a call are assembled.
- An EDR is emitted and rated
- The call is kept open for more CDRs.

When the next semaphore is sent in, all CDRs that have arrived since the last semaphore are assembled as an EDR and rated.

By default, FCT_CallAssembling calculates a call's time duration by subtracting the end time of the last CDR that has arrived from the start time of the first. Because the CDRs between the first and last are not used in the calculation, it does not matter if they have arrived when the EDR is created. If a CDR between the first and last CDRs is missing when the time duration is calculated, the missing CDR is dropped when it finally does arrive.

Example time duration registry

```
CallAssembling
 {
  ModuleName = FCT_CallAssembling
  Module
   {
      Active            = True
      AssembleVolume    = False
      AssembleSGSN      = False
      SplitAtGaps       = False
      MaxDuration       = 900
      Path              = ./data/assy
      FileName          = calls
      Mode              = Normal
      RejectMissingChain = True
      CallDurationTolerance = 59
      DropLateCDRs          = False

      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
      }
   }
 }
```

> **Note:**
>
> If you use both the FCT_CallAssembling and FCT_Reject in the same pipeline, use the FCT_Reject module **CallAssemblingModule** registry entry to ensure that complete EDRs are recycled. See "FCT_Reject".

Rating Calls by Volume of Data Sent

Use the **AssembleVolume** and/or **AssembleSGSN** registry entries to direct FCT_CallAssembling to rate calls by the volume of data sent. Both **AssembleVolume** and **AssembleSGSN** ensure that you capture the entire volume of data sent for a single call, and they both direct FCT_CallAssembling to rate a call only after all of its call records have arrived.

If you do not plan to rate calls by volume of data sent, leave **AssembleVolume** set to **False**. This saves system resources by disabling the registry entries which rate by volume.

The volume-based rating entries protect against lost revenue. Call records can arrive in any order, so it is not unusual for an intermediate segment to arrive after the first and last segments have arrived and been rated. In this case, any intermediate segments that arrive after the call is closed are dropped, and all the volume they contain are lost (and so is the revenue).

By default, FCT_CallAssembling calculates the time duration for each CDR *individually*. FCT_CallAssembling subtracts the end times from the start times of each CDR to calculate the time duration. This module then adds the time durations of all CDRs as they arrive to create a grand total for the call.

If your business requires that all non-contiguous CDRs be rated as separate EDRs (TAP requires this) set the **SplitAtGaps** registry entry to **True**. If not, then set this entry to **False**, and non-contiguous CDRs will be collected into a single EDR, saving system resources.

However, if you are rating calls by volume of data sent, lost intermediate call records will cause calls to remain open indefinitely. Send in a **Flushlimit** entry frequently to avoid this.

For information on data calls that you want to keep open indefinitely and rate periodically, see "Rating Continuous Data Calls by Segment".

Example volume of data call registry

```
CallAssembling
 {
  ModuleName = FCT_CallAssembling
  Module
   {
      Active            = True
      AssembleVolume    = True
      AssembleSGSN      = True
      SplitAtGaps       = False
      Path              = ./data/assy
      FileName          = calls
      RejectMissingChain = True
      CallDurationTolerance = 59
      DropLateCDRs          = False

      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
      }
   }
 }
```

Example TAP volume of data call registry

This example adheres to the TAP 3.10 standard. Volume and SGSN data are recorded precisely at the start and end times of each call segment. Because **SplitAtGaps = True**, if any call segments are missing, the segments before and after it are emitted as separate EDRs.

```
CallAssembling
 {
  ModuleName = FCT_CallAssembling
  Module
   {
      Active            = True
      AssembleVolume    = True
      AssembleSGSN      = True
      SplitAtGaps       = True
      Path              = ./data/assy
      FileName          = calls
      RejectMissingChain = True
      CallDurationTolerance = 59
      DropLateCDRs          = False

      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
      }
   }
 }
```

Specifying a Time Error

Use the **CallDurationTolerance** startup registry entry to specify an amount of time (in seconds) that is an acceptable amount of time error for each call.

> **Note:**
>
> The default value for this entry correctly handles most calls, so you do not need to change this entry unless you notice a problem.

Mobile phone calls are commonly split into multiple call records. Each call record should start at the same second that the previous one ended. For example, a call record with an *end* time of 12:01:30 should be followed by the next call record with a *start* time of 12:01:30. Unfortunately, it is common for these start and end times to either not quite match, or to overlap slightly (time error). The reason may be that the call records come from different routing switches with clocks that have not been synchronized, or the switches themselves have difference time tolerances.

The **CallDurationTolerance** default value is 60 seconds of tolerance to compensate for this time error. If a call has less than 60 seconds of error, a call is considered complete and sent for rating. Otherwise the call is left open and must be closed with a semaphore entry. A single call with 10 call records, each overlapping by 3 seconds, creates a call with 30 seconds of time error. This 30-second time error is less than the **CallDurationTolerance** 60-second time limit, so the call is considered complete and sent down the pipeline for rating.

> **Note:**
>
> Setting this entry too low causes an inordinate number of calls to be left open indefinitely. Setting this entry too high can cause the pipeline to rate calls with missing call segments. The 60-second default value is appropriate for most BRM implementations.

Rating Continuous Data Calls by Segment

Use the **KeepCallOpen** semaphore registry entry together with **FlushLimit**, to keep calls open indefinitely, and rate them in segments periodically. This feature is designed for data calls that are kept open continuously (days at a time). These long data calls are usually rated periodically to capture revenue.

KeepCallOpen is an update registry entry sent in the **FlushLimit** semaphore.

For example, a bank might keep a continuous call open to each of its ATMs to pass data back and forth. Using the default behavior, the call would not be rated for days. You will probably want to capture the revenue for these types of calls periodically, perhaps every 12 or 24 hours. Setting **KeepCallOpen** to **True** keeps, these calls open. You then rate these calls in segments by sending in a **FlushLimit** semaphore entry.

If you set **KeepCallOpen** to **True** and send it in with the **FlushLimit** semaphore every 12 hours, an EDR is created twice a day, each EDR rating the previous 12 hours of call time and volume.

KeepCallOpen is an entry in the **FlushLimit** update registry entry.

By default **KeepCallOpen** is set to **False**. The default behavior directs this module to rate the call when the first and last segments have arrived, or when the **FlushLimit** is sent, whichever comes first. Any subsequent records are ignored.

Rating Partial Calls by Service

Use the **FlushServiceCode** semaphore registry entry to limit the effect of **FlushLimit** to calls with specific service codes. **FlushServiceCode** is sent as an entry in the **FlushLimit** update registry entry. If the **FlushServiceCode** entry matches the value of the DETAIL.INTERNAL_SERVICE_CODE field of the call segment, the call is flushed. All other partial calls are ignored.

For example, the following **FlushServiceCode** entry directs the pipeline to rate only calls with the **dat** service code:

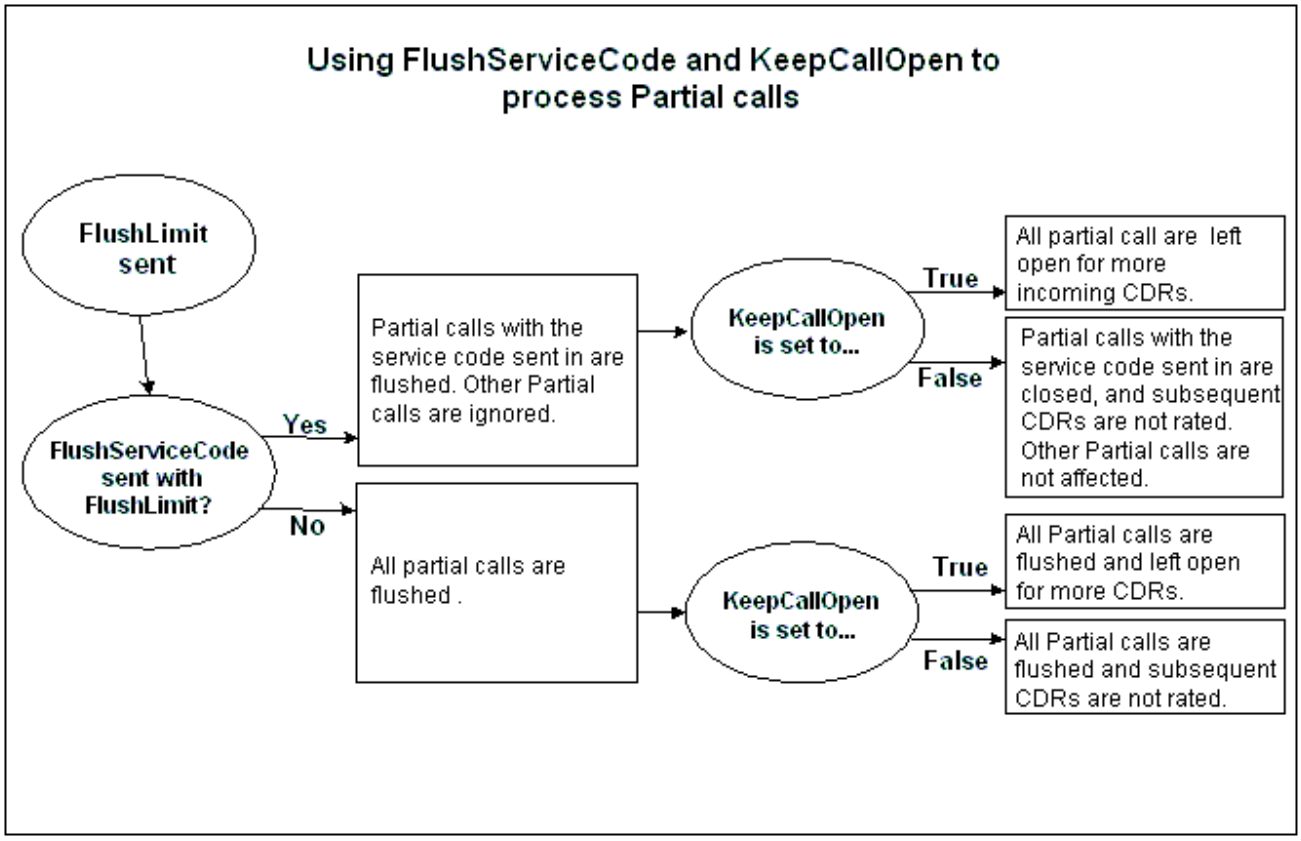
```
{

  FlushLimit=0
  FlushServiceCode = dat

}
```

Figure 5-1 shows how the **FlushServiceCode** and **KeepCallOpen** semaphore registry entries interact.

Figure 5-1 FlushServiceCode and KeepCallOpen Semaphore Interaction



Capturing Fields From the Last Call Record

Use the **AssembledEDR** startup registry entry to capture data from **Basic Detail Record** fields for the **L** call segment.

The final record of a call has a long duration indicator of **L** (last), however the **L** call record may not be the last to arrive at the pipeline. In some cases it is important that information be captured from the **L** call record. For example, the number that terminated the call can only come from the **L** call record.

If **AssembledEDR** is used, the pipeline captures **Basic Detail Record** data from the **L** call record and adds it to the EDR emitted, regardless of whether it was actually the last call record received.

List any **Basic Detail Record** fields to capture data from in the **AssembledEDR** entry, including any custom fields that your business requires. The pipeline will then include the data from those fields on the EDR that is compiled and processed. See "Sample Registry" for an example.

Tracking the Status of Assembled Calls

You can use a semaphore command to receive a report about the status of calls currently being assembled. The report provides the following information:

- The number of partially assembled calls.

- The number of EDRs currently waiting to be assembled.
- The number of late EDRs that are parts of calls that are no longer being assembled, for one of the following reasons:
  - The call was assembled and sent to the pipeline. In this case, the first and last portion of a call was processed, and an intermediate part arrived after the EDR had been sent into the pipeline.
  - The call was flushed by a **FlushLimit** semaphore, and another portion arrived after the call was flushed.

  The report includes the following data about late EDRs:
  - The number of late-arriving EDRs for flushed calls.
  - The number of late-arriving EDRs for assembled calls.
  - The total number of late-arriving EDRs.

In addition to creating the report, each assembled EDR includes a NUMBER_OF_CDRS field that stores the number of CDRs that were included in the EDR. You can use this data in an aggregation scenario to gather additional data about assembled calls.

> **Note:**
>
> For EDRs that are not part of split calls, the FCT_CallAssembling module enters **1** in the NUMBER_OF_CDRS field.

Migrating Call Assembling Data Between Releases and Pipelines

When you upgrade BRM from one release to another or apply patches, you need to migrate the call data in your **.dat** files to the new format by using the XML support provided with the FCT_CallAssembling module.

You use the following semaphore registry entries to change the format of the data from one release or pipeline to another:

- **ExportDataToXml**
- **ImportDataFromXml**

For more information, see the semaphore file entries in "FCT_CallAssembling".

To migrate call assembly data, perform the following steps:

1. Export the data from your existing data files to an XML file by using a semaphore registry file with the **ExportDataToXml** entry:

```
ExportDataToXml
{
   CallsPerFile = Value
}
```

2. Import the data from the XML file into the data file with the new format by using the **ImportDataFromXml** entry:

```
ImportDataFromXml
{
   FileName = filename.xml
}
```

Assembling Calls with Multiple Pipelines

When you use the FCT_CallAssembling module, you must process the EDRs for each account in the same pipeline. If you assemble calls in multiple pipelines for the same account, the call segments may get processed by different pipelines and cannot be assembled.

Discarding and Skipping EDRs

You can use the FCT_Discard module to skip or discard EDRs:

- Skipping an EDR removes it from the pipeline.
- Discarding an EDR sends it to a different output stream. You will probably want to audit the information in EDRs with a LONG_DURATION_INDICATOR of **XC**, **XO**, or **XP** before you discard them.

In both the cases the state of the EDR becomes invalid. (To indicate a discarded EDR, a value is entered in the DETAIL.DISCARDING field.)

For example, you can filter the following EDRs:

- Discard EDRs that are older than three days and have a B_NUMBER that begins with 0049.
- Discard EDRs that have a RECORD_TYPE that begins with a 9 followed by two digits, an INTERN_SERVICE_CODE that ends with a 2, and a WHOLESALE_CHARGED_AMOUNT_VALUE of 0.

For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

To create a valid mapping, the data in the EDR must match with all of the mapping data.

Configuring EDR Discarding

To configure EDR discarding:

1. Specify which EDRs to discard or skip. See "About Configuring Discard and Skip Expressions".
2. Configure the OUT_DevNull module as the output stream for discarded EDRs. See "Configuring Output of Discarded EDRs".
3. Configure the FCT_Discard module. See "FCT_Discard".
   - Use the FCT_Discard module **StreamName** registry entry to specify the output stream.
   - To send EDRs to different output streams without making them invalid, use the FCT_EnhancedSplitting module. To process EDRs that belong to accounts in different BRM database schemas, see "FCT_AccountRouter".

About Configuring Discard and Skip Expressions

To specify which EDRs to discard or skip, you create a set of regular expressions using a set of data fields. These fields are not mandatory so if they do not exist in the EDR description it does not matter. If all expressions match the available fields, the EDR will be removed.

You can create different discard rules for separate pipelines.

You define discard rules in Pricing Center or Pipeline Configuration Center (PCC). The discard rules are stored in the IFW_DISCARDING table.

Use the **Reload** semaphore file entry to reload regular expression patterns after you change them.

You can discard or skip EDRs based on the following:

- Rank. This specifies the order in which to evaluate the rules.
- Record type.
- Source network.
- Destination network.
- Call complete indicator.
- Long duration indicator.
- Usage class.
- Internal service code.
- GSM switch or GPRS switch.
- Tariff class.
- Tariff subclass.
- Connection type.
- Connection subtype.
- B number.
- The age of the EDR.
- If the wholesale charge should be zero.

For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

To configure the FCT_Discard module. See "FCT_Discard".

Configuring Output of Discarded EDRs

Use the OUT_DevNull module to handle EDRs that should be discarded from a pipeline.

To discard EDRs, do the following:

- Configure the FCT_Discard module. See "Discarding and Skipping EDRs ".
- In the pipeline, configure the OUT_DevNull module. See "OUT_DevNull".

OUT_DevNull is a submodule of the Output Collection module. All registry parameters and error messages are handled by the Output Collection module. See "Output Collection".

Generating Multiple TAP MOC and MTC Records

When you process TAP files, you use the ISC_TapSplitting iScript to splits mobile originating and terminating EDRs into multiple EDRs when the CDR contains more than one basic service. The ISC_TapSplitting creates a new EDR for each additional basic service.

Splitting mobile originating and terminating EDRs enables CDR rejection when a basic service record associated with a CDR is in error. It also permits custom validations to be added prior to splitting.

Service information for all secondary services (supplementary, VAS, CAMEL), which are part of a basic service EDR, are added to the last basic service EDR. Charge information for these secondary services are added to the last charge breakdown record of the last basic service EDR.

EDR splitting is performed after TAP validation. TAP files and EDRs that are rejected due to errors are not split. After the new EDRs are created, the original EDR is deleted.

To configure ISC_TapSplitting, see "ISC_TapSplitting".

Using Rules to Send EDRs to Different Output Streams

> **Note:**
>
> To send EDRs to different output streams for the Rated Event (RE) Loader, use the IRL_EventTypeSplitting iScript. See "Sending EDRs to Pipeline Output Streams".

Use the FCT_EnhancedSplitting module to specify different output streams for EDRs based on rules. For example:

- You can split EDRs for different service types into different output streams.
- You can split EDRs from roaming outcollects and incollects into different streams.

To send EDRs to different output streams, you define a set of rules. For example, you can send all telephony EDRs from a specific network to a different output stream.

You can use the following data in the rules:

- Record type.
- Service code.
- Usage class.
- Source and destination network.
- Switch.
- Trunk in/out.
- A number and B number area code.
- Normalized C number area code (forwarded or routed number).

This example assigns all telephony EDRs with an A number starting with 49 to the Out49 system brand:

```
Service code :   TEL
A number:        0049.*
System brand:    Out49
```

In addition you can specify the order in which to evaluate the rules, and the dates when the rule is valid.

The FCT_EnhancedSplitting module evaluates each EDR against the rules. The first rule that matches the criteria defines the system brand to use. The system brand is identified by a code. You use that code to map the system brand to an output stream in the registry. In this example, system brand **Out49** is mapped to the **EdrOutputOut49** output:

```
SystemBrands
{
Out49 = EdrOutputOut49
}
```

You can create separate splitting rules for different pipelines.

You can use a semaphore file entry to reload a new set of rules.

Configuring Enhanced Splitting

To configure enhanced splitting, you do the following:

1. Configure system brands. Each system brand is mapped to an output stream.
2. Use Pricing Center or Pipeline Configuration Center (PCC) to configure the splitting rules. Each rule is associated with a system brand. If an EDR matches a rule, it uses the system brand defined in the rule.

   To create a valid mapping, the data in the EDR must match with all of the mapping data.

   For the data fields, you use regular expressions. For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".
3. Configure an output stream for each system brand. See "Configuring EDR Output Processing".
4. Configure the FCT_EnhancedSplitting module. This defines the output stream for each system brand. See "FCT_EnhancedSplitting".

Sending EDRs to Pipeline Output Streams

When you load rated events in the BRM database, RE Loader loads events for separate services from separate directories. Therefore, your pipeline needs to send EDRs to separate output streams for each internal service code. To do so, you use the IRL_EventTypeSplitting iScript.

To specify the output stream, you edit the **IRL_EventTypeSplitting.data** file. This file maps service codes to output streams. For example, this entry maps the TEL service code to the TelOutput stream:

```
TEL;TelOutput
```

> **Note:**
>
> You can also use the FCT_EnhancedSplitting module to send EDRs to different output streams based on EDR content. See "Using Rules to Send EDRs to Different Output Streams".

The IRL_EventTypeSplitting iScript must run after the FCT_ServiceCodeMap module and before FCT_Reject module.

To configure the IRL_EventTypeSplitting iScript, you configure it as part of the FCT_IRules module. See "FCT_IRules".

For more information, see "IRL_EventTypeSplitting".

Using Pipeline Manager with Multiple Database Schemas

In a multischema environment, you start a separate instance of Pipeline Manager for each BRM database schema.

- Use the FCT_AccountRouter module to find the account and send the EDR to the correct instance of Pipeline Manager. The FCT_AccountRouter runs in its own instance of Pipeline Manager.
- Use the DAT_AccountBatch module to supply data to the FCT_AccountRouter module. It runs in the same instance of Pipeline Manager as the FCT_AccountRouter module. See "DAT_AccountBatch".

To find the A number customer and the B number customer, the FCT_AccountRouter module does the following:

1. The module looks for the following data in the EDR:
   - The internal service code that indicates which data can be used to identify the account. For example, if the internal service code is a telephony service, the identifying data is the A number. A different service might use the IMSI as the identifier.

     You identify which data to use by using Pricing Center or Pipeline Configuration Center (PCC).
   - The timestamp for the EDR. The timestamp is important because telephone numbers can be used by different accounts at different times.
2. The module uses the DAT_AccountBatch module to look up the account. See "DAT_AccountBatch".

   Note:

   - If no A customer is found, the EDR is rejected. If the B customer is missing, no error is generated.
   - Because phone numbers can be recycled, the search is made on data from BRM audit objects.
3. The DAT_AccountBatch module returns the database number.
4. The FCT_AccountRouter sends the EDR to the correct pipeline, using the configuration defined in the registry.

Setting Up Account Identification in Multischema Systems

To set up account identification in a multischema system, do the following:

1. Configure service and account data in the Pipeline Manager database:
   - Configure internal service codes. The DAT_AccountBatch module retrieves account information based on which services are rated.

- Configure how the FCT_AccountRouter module looks up accounts (for example, by telephone number or IMSI). You specify which data is used for identifying accounts when you configure the FCT_Account module.
- (Optional) Configure account ID prefixes to use for handling duplicate telephone numbers. You configure this when you configure the FCT_Account module.

2. Configure the FCT_AccountRouter. See "FCT_AccountRouter".

> **Note:**
>
> Since the FCT_AccountRouter module needs the internal service code, the FCT_AccountRouter module must be placed after the FCT_ServiceCodeMap module in the pipeline.

3. Configure the DAT_AccountBatch module **UseAsRouter** registry entry.

   If set to **True,** the module is used by the FCT_AccountRouter module to route EDRs to separate Pipeline Manager instances. See "FCT_AccountRouter".

   If set to **False** (the default), the module is used by the FCT_Account module to get account data.

   See "DAT_AccountBatch".

6

Setting Up EDR Enrichment

This chapter describes ways to enrich event data record (EDR) data for rating by the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Identifying the Network Operator/Service Provider

If your network supports multiple network operator/service providers (NO/SPs), you can use the FCT_NOSP module to identify the various NO/SPs. This module uses the source and destination codes and A number prefix in the EDR to assign a new source and destination code for the NO/SP.

Use this module when you need to identify the NO/SP and the NO/SP information is not available in the EDR; for example, when mobile networks are separated by means of the A number and you need to segment calls.

To map NO/SP data, you do the following:

1. Use Pricing Center or Pipeline Configuration Center (PCC) to create NO/SP maps. See "Creating an NO/SP Map".

   You can also use a file to configure the NO/SP map.
2. Configure the FCT_NOSP module. See "FCT_NOSP".
3. Configure the DAT_NOSP module. See "DAT_NOSP".

   If you store NO/SP data in a file, see "Creating an NO/SP Data File".

Creating an NO/SP Map

To create NO/SP mappings, you use Pricing Center or Pipeline Configuration Center (PCC) or a text file to set up the data that specifies how to map NO/SPs.

The mappings are based on the following data:

- Map group.
- The order in which mappings are applied. The first NO/SP map that matches is used.
- The source network and network destination recorded in the CDR.
- The phone number prefix to match.
- The new network source and destination to use in the EDR.

For information on the regular expressions you can use, see "About Using Regular Expressions when Specifying the Data to Extract".

Pricing Center or Pipeline Configuration Center (PCC) stores the mappings in the IFW_NOSP database table.

For information on using a file to specify NO/SP maps, see "Creating an NO/SP Data File".

Creating an NO/SP Data File

You can store data for the DAT_NOSP module in the Pipeline Manager database or in a text file.

The configuration file must be an ASCII file. Each row defines one mapping rule. All fields in a row are separated by a semicolon (;). The fields in one row have the following order:

1. MAPGROUP (NOT NULL)
2. RANK (NOT NULL)
3. OLD_SOURCE
4. OLD_DESTINATION
5. A_PREFIX
6. NEW_SOURCE (NOT NULL)
7. NEW_DESTINATION (NOT NULL)

Setting Up Social Numbers

In some cases, customers want a B number to not appear on an invoice. For example, some countries require that certain called numbers remain anonymous, such as the number for a treatment center. You can set up social numbers to hide specific B numbers.

The social flag functionality is run by the FCT_SocialNo module. When it finds a social number, the module sets a flag in the EDR B_MODIFICATION_INDICATOR field. You can use this flag to customize how to handle the number; for example, remove the last three digits or not allow the EDR to be included in an invoice.

To set up social numbers:

1. Use Pricing Center or Pipeline Configuration Center (PCC) to specify social numbers.

   To specify social number, you specify the number that identifies it internally and give the number a descriptive name.

   The FCT_SocialNo module looks for an exact match of this number in the EDR.

   You can also use a file to specify social numbers. See "Creating a Social Number Data File".
2. Configure the FCT_SocialNo module. See "FCT_SocialNo".

Creating a Social Number Data File

You can specify social numbers in Pricing Center, Pipeline Configuration Center (PCC), or a file.

The social number data file uses the following syntax:

```
number1
number2
...
```

The number uses the same format as the normalized B number in the EDR.

International-access-code c\Country-code National-destination-code Access-code

For example:

```
0014085555555
```

Creating Call Destination Descriptions

You can set up descriptions for call destination area codes. For example, a prefix of 001408 can be described as "San Jose, California." The description is displayed on the customer's bill.

Prefix/description mapping is performed by the FCT_PrefixDesc module. You set up a mapping between prefixes and descriptions. The module finds the best match for the prefix and adds the description to the EDR DESCRIPTION field.

To set up prefix/description mapping:

1. Map prefixes and descriptions in Pricing Center or Pipeline Configuration Center (PCC).

   > **Note:**
   >
   > You can also use a text file. See "Creating a Prefix/Description Data File".

2. Configure the FCT_PrefixDesc module. See "FCT_PrefixDesc".
3. Configure the DAT_PrefixDesc module. See "DAT_PrefixDesc".

Setting Up Prefix/Description Mapping

To set up prefix/description mapping in Pricing Center or Pipeline Configuration Center (PCC), enter the following:

- The area code prefix. The prefix must have the same format that is used for normalization within a pipeline. The FCT_PrefixDesc module does not normalize numbers.
- The prefix type:
  - National
  - International
  - Special

  > **Note:**
  >
  > If you use a file to store the mappings, you do not enter any prefix type. See "Creating a Prefix/Description Data File".

- The description to use, for example, "New York City."

  > **Note:**
  >
  > All prefix/description pairs must be unique.

Creating a Prefix/Description Data File

You can define prefix/description mappings in the Pipeline Manager database, or in a file.

The mapping file has the following structure:

```
Prefix1; Description1
Prefix2; Description2
```

Mapping Multiple Phone Numbers to a Single Number

Customers with more than one telephone number sometimes want to get only one bill for all their numbers. You can map multiple telephone numbers to one number. This mapping is performed by the FCT_CliMapping module. The module uses the A number to search for a mapping entry. If there is a mapping entry, the A number is replaced by the Caller Line Identification (CLI) number

For example, a customer with a primary number 14085722000 could have five extra telephone numbers but would like to be billed as if they originated from the same number. To accomplish this for all the calls originating from the other five numbers, the A number is mapped to 14085722000 in the EDR.

To configure CLI mapping:

1. Define the mapping in a file. See "Creating a CLI Mapping File ".
2. Configure the FCT_CliMapping module. See "FCT_CliMapping".

Creating a CLI Mapping File

As shown in Table 6-1, the FCT_CliMapping module requires an ASCII mapping file that contains the numbers to map to a single number.

- Every new line defines a mapping.
- Fields are separated by semicolons (;).
- There should be no semicolon at the end of a line.

Table 6-1 FCT_CliMapping Module File Structure

Column	Name	Position	Length	Format	Description
1	CLI_FROM	1	25	X(25)	Start CLI
2	SERVICE_CODE	27	5	X(5)	Service Code
3	MAPPING_CLI	33	25	X(25)	Mapping CLI
4	CUST_NUMBER	59	10	X(10)	Customer number
5	SUBSC_NUMBER	70	20	X(20)	Subscriber number
6	ISDN_RANK	91	1	X(1)	Rank of the ISDN number

Managing Number Portability

Customers may want to change network operators but retain their existing phone number. You can maintain number portability data to keep a track of the network operator a customer uses.

For maintaining number portability data, the DAT_NumberPortability module loads the data from the number portability file into the memory (pipeline's run-time memory).

The FCT_NumberPortability gets the number portability data from the DAT_NumberPortability module. The data includes the CLI, the new network operator ID, and the date that the customer changed network operators. The FCT_NumberPortability module uses the date of the event to determine which network operator applies. If the new network operator applies, FCT_NumberPortability module updates the new network operator ID in the EDR.

BRM's number portability feature support batch pipeline (offline mode). See "Number Portability for the Batch Pipeline".

For batch pipeline rating, the FCT_NumberPortability module updates the EDR with the appropriate network operator ID based on the time stamp when the network operator changed.

For real-time pipeline rating, the FCT_NumberPortability module updates the EDR with the appropriate network operator ID based on the time stamp when the network operator changed. The FCT_Opcode module creates an opcode input flist containing the new network operator information. The flist is used for rating the calls in real time.

Number Portability for the Batch Pipeline

BRM uses semaphores to load the number portability data file using the batch process in the pipeline.

The number portability process for the batch pipeline makes the system inactive when the new number portability records are updated in the system.

About Number Portability Files

The number portability file is an ASCII file that stores the number portability data. The data from these files is loaded into the DAT_NumberPortability module.

You enter the following information in the number portability file:

- CLIs.
- The time stamp when CLIs are ported to a new network operator.
- The new network operator ID.

See "Creating a Number Portability Data File".

The following example shows a sample of number portability records in a number portability file:

```
Call Line ID            Date              Network Operator ID
408555                  20080101000000    D030
408555                  20080110000000    D010
```

The records in the example show that the subscriber changes the network operator to D030 on January 01, 2008. From January 01 to January 1, the subscriber is with the network operator D030. On January 10, the subscriber changes the network operator to D010.

BRM uses the following number portability files:

- Primary number portability file: Contains the primary number portability records that already exist.
- Delta number portability file: Contains the additional number portability records that need to be added to the memory and the primary number portability file.

You use the primary number portability file when you use the **reload** probe or the **Reload** semaphore to reload number portability data.

You use the delta number portability file when you use the **deltaLoad** probe to update the primary number portability file.

To manage number portability data, you can do the following:

- Purge data from the primary number portability file. To purge data from the file, use the purge utility (**purge.np_data.pl**). See "Purging and Reloading the Memory Records".
- Reload the data from the primary number portability file to the memory. See "Purging and Reloading the Memory Records".
- Append the data from the delta number portability file to the memory and subsequently to the primary number portability file. See "Appending Additional Number Portability Records".

Creating a Number Portability Data File

The DAT_NumberPortability module reads the required data from a reverse-sorted ASCII file.

Each row of the number portability file contains a CLI number, a date and time (when the numbers are ported), and a network operator ID as shown in Table 6-2.

Table 6-2 Input File for DAT NumberPortability module

Column	Format	Description
Call Line Identity Number	String	Specifies the telephone number or a part of the telephone number.
Portation date and time	*YYYYMMDDhhmmss*	Specifies the date and time of the number portation.
Network Operator ID	D*xxx*	Specifies the new telephone carrier.

An example of a sample file:

```
089761   20020910101230   D030
089761   20020912101230   D018
089545   20020820084000   D017
089545   20020920230010   D030
```

> **Note:**
>
> The columns are separated by spaces.

Use the DAT_NumberPortability FileName registry entry to specify the file name for the number portability data file.

Purging and Reloading the Memory Records

The purge utility is used to delete existing records (from the number portability file) that are older than a time stamp specified in the utility. You can purge the data from the memory by using the purge utility. For purging the records from the memory, you must first purge records

from the number portability file and then use the **reload** probe to ensure that the records in the number portability file and the memory are in sync.

To purge and reload number portability records:

1. Run the **purge_np_data.pl** utility to purge the primary number portability data file:

   ```
   purge_np_data.pl NP_FileName TimeStamp [-b backup_filename] [-n]
   ```

   where,

   - *NP_FileName* specifies the name of the primary number portability file. For example, **Primary_NP_Data.data**.
   - *TimeStamp* specifies the date prior to which all the number portability records are purged. For example, 20080105000000.
   - **-b** *backup_filename* specifies the name of the backup file that will contain the unpurged number portability records. For example, **Primary_NP_Data.bak**.

   This command takes a backup of the existing number portability records in the backup file and deletes all number portability records prior to the date (timestamp) specified from the primary number portability file.

2. Initiate the Reload semaphore to reload the purged number portability data into the memory.

   This ensures that the records in the number portability file and the memory are in sync.

   > **Note:**
   >
   > For the sample entry of **Reload** semaphore, see "Sample Semaphore File Entry".

   If the reload operation fails, the memory data will contain all the unpurged data. The primary number portability data file is moved to a *pipeline_home***/npdata/error** directory. *pipeline_home* is the directory where you installed Pipeline Manager.

Appending Additional Number Portability Records

You can use the delta number portability file to append additional or newly ported records to the primary number portability file. The delta number portability file contains the additional records that are in the same format as the primary number portability file. By default, the delta number portability file is stored in the *pipeline_home* directory. You can also manually add the additional records in the primary number portability file but this process is cumbersome when you need to add many records.

To append additional number portability records:

1. Initiate the **AdditionalNumPortData** semaphore with the delta number portability file name as a value. The delta number portability file contains additional entries in the same format as the number portability file.

   The additional entries are first added to the memory and then the memory data is dumped into the primary number portability file so that the records in the memory and the primary number portability file are in sync.

> **Note:**
>
> For the sample entry of **AdditionalNumPortData** semaphore, see "Sample Semaphore File Entry".

2. Initiate a **PrintData** semaphore to print the records to a file.

   All number portability data in the memory is copied to the file. By default, this file is created in the *pipeline_home* directory.

> **Note:**
>
> For the sample entry of the **PrintData** semaphore, see "Sample Semaphore File Entry".

Setting Up Number Portability

You must set up number portability to maintain the number portability data to keep track of the network operator a customer is using. You can configure number portability for the following:

- Batch pipeline
- Real-time pipeline

Setting Up Number Portability for Batch Pipeline

To set up number portability for the batch pipeline:

1. Configure the FCT_NumberPortability module. See "FCT_NumberPortability".
2. Configure the DAT_NumberPortability module. See "DAT_NumberPortability".

   When you configure the DAT_NumberPortability module, you specify the following:

   - The search method. See "Configuring Number Portability Search".
   - Normalization. See "Configuring Normalization for Number Portability".
   - The number portability data file. See "Creating a Number Portability Data File".

Setting Up Number Portability for Real-Time Pipeline

To set up number portability for the real-time pipeline:

1. Configure the FCT_NumberPortability module. See "FCT_NumberPortability".
2. Configure the DAT_NumberPortability module. See "DAT_NumberPortability".

   When you configure the DAT_NumberPortability module, you specify the following:

   - The search method. See "Configuring Number Portability Search".
   - Normalization. See "Configuring Normalization for Number Portability".
   - The number portability data file. See "Creating a Number Portability Data File".
3. Configure the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript in the registry file and place it in the processing pipeline after the FCT_NumberPortability module. See "ISC_PopulateOpcodeandUtilBlock_Diameter".

4. Configure the ISC_MapNetworkOperatorInfo iScript and place it after the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript in the registry file. See "ISC_MapNetworkOperatorInfo".
5. Configure the FCT_Opcode module and place it in the processing pipeline after the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript. See "FCT_Opcode".

Configuring Number Portability Search

You can configure which of the following search methods the DAT_NumberPortability module uses to find a phone number's current network operator:

- **Best match** searches the number portability file for objects with the best combination of matching phone number prefix and recent port date. For example, a number portability file includes the following entries and a pipeline module receives a CDR with a date of November 2 and the phone number 4085551234:

  ```
  Call Line ID      Date              Network Operator ID
  408               20081001000000    D030
  408555            20080801000000    D029
  408555            20080708000000    D028
  ```

  The DAT_NumberPortability module returns network operator D029 because the entry contains the most recent port date with the most matching prefix numbers.
- **Exact match** searches for the first object that exactly matches a given phone number.
- **Any prefix match** searches for the first object with a matching prefix.

To configure the search method, use the DAT_NumberPortability **SearchMethod** registry entry. See "DAT_NumberPortability".

Configuring Normalization for Number Portability

Use the following DAT_NumberPortability registry entries to specify number normalization data:

- CountryCode
- NationalAccessCode
- InternationalAccessCode
- InternationalAccessCodeSign

For example, you can normalize this phone number:

04106760279

so that it becomes:

00494106760279

See "DAT_NumberPortability".

7

Setting Up Pipeline Aggregation

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager aggregation feature.

About Aggregation

You use the aggregation feature to compile data from event data records (EDRs). Aggregation is typically used for the following:

- To filter and summarize data for compiled statistics about service usage, customer activity, network activity, dealer commissions, and so forth. For example, you can compile the sum of charges per customer per month.
- To aggregate data and use for rating. For example, you can compile GPRS usage records and rate the aggregated amount.
- To output a back-out file used as input for rerating.

About Setting Up Aggregation Pipelines

Because the results of aggregation are typically not used by other modules, and because the aggregation process uses more resources than rating, aggregation is typically performed by a separate pipeline. You can run an aggregation pipeline parallel to a rating pipeline, so that the same input files will be processed by both pipelines.

The aggregation pipeline processes EDRs in transactions, typically one transaction per input file. When the transaction is finished, the data is written from memory to a file.

After every transaction, the aggregated results are written to a file that contains the transaction ID. For every transaction and aggregation scenario a result and a control file is created. The control files are used by the Database (DB) Loader utility to load the results into the database. See "Creating Aggregation Scenarios".

You define the control file in the FCT_AggreGate registry entries. After the data is processed by the pipeline, you use DB Loader to load the data from the file to a database. This can be the same database used by Pipeline Manager or a different database. You can run the utility automatically or manually. See "Database Loader".

See "Creating Aggregation Scenarios".

See "FCT_AggreGate", "DAT_ScenarioReader".

> **Note:**
>
> If you configure aggregation for rerating, ensure that the FCT_AggreGate module is working in back-out mode.

About Aggregation Scenarios

You aggregate data by creating aggregation scenarios (see "Creating Aggregation Scenarios"). An aggregation scenario specifies which data to use and how to process the data. You use the following criteria when setting up aggregation scenarios:

- **Filter conditions**: Allows you to choose which EDRs to aggregate; for example, you can specify all calls for a single customer, service, time period, or charge.

  > **Note:**
  >
  > Conditional filters for combined attributes are not supported.

- **Grouping rules:** Allows you to group calls according to various criteria; for example, you can aggregate sums based on levels of duration, as follows:
  - All calls less than one minute.
  - All calls from one to five minutes.
  - All calls over five minutes.
- **Aggregation functions:** You can do the following with the data:
  - Count the number of selected events.
  - Sum the values of selected events.
  - Generate the square sum from values of selected events.

You can use any EDR field for filtering, grouping, or aggregating. However, aggregation criteria are limited to the following:

- The data must be in the EDR.
- The data must be available for an overall aggregation, typically of a NUMBER type.

> **Note:**
>
> An aggregation scenario is analogous to an SQL SELECT statement with a GROUP BY claus.
>
> For example, select all EDRs where the time period is peak time, then group the calls by levels of duration (for example, 0-1 minute, 1-5 minutes), and aggregate the charges for each level of duration.

Creating Aggregation Scenarios

When you create an aggregation scenario, you define filter, grouping, and aggregation functions. When you run an aggregation pipeline, you specify which scenario to use in the FCT_AggreGate registry.

Defining Filter Criteria

Define the following when defining filter criteria for a scenario:

- The status of the filter criteria (active or inactive).
- The EDR field that the filter is applied to.
- The data type to use; for example, string, number, or date/time.
- The value to filter with.
- The function used for aggregating:
  - Count
  - Sum
  - Square sum
- The decimal precision for the aggregated result.

Specifying Scenario Attributes

In addition to defining the filters, groups, and aggregation functions, you can use the registry to define the following attributes for each scenario:

- The EDR container description for the scenario.

> **Note:**
>
> You can only set up filters and groups for fields included in the EDR container description.

- A default table name where the aggregation results are stored.
- The maximum number of aggregations held in memory before writing the data to a file. Enter **0** to specify an infinite number.
- Default directories for the following:
  - Where to store the aggregation results file (temporary and finished data).
  - Database (DB) Loader control files.
- A default delimiter for the single values of grouping and aggregation fields.

About Creating Groups

You use grouping rules to group and order aggregation results. Grouping rules are applied to EDR container fields and associated with a rank. For example, aggregation results can be grouped first by the event originator, then by zones, and then by time periods, or by duration, area code, or geographic zone.

When you define groups, you define the database columns that contain the aggregated data. For example, if you group aggregation results by duration, a column for that value is created in the table in the results file.

A group includes the following attributes:

- The EDR field for which the values are grouped; for example, time period or duration.
- The column name where the aggregations for the group are written in the results file.
- The data type for the grouped data:
  - String

  - Number
  - Date/time
- If applicable, the output format for the data. This can be date/time values or string format; for example UPPER, LOWER or SUBSTR(x,y).

About Creating Classes for Groups

You create classes to group results for a type of data. For example, to group aggregation results by the duration of events, you can create the duration classes shown in Table 7-1:

Table 7-1 Duration Classes

Class	Value to group by
Duration Class A	0 to 60 seconds
Duration Class B	61 to 300 seconds
Duration Class C	301 to 600 seconds
Duration Class D	More than 600 seconds

When creating a class, you define the following categories:

- The name; for example, duration.
- The data type:
  - String
  - Number
  - Date/time
- Class items. For each class item you define the following:
  - The name; for example, duration.
  - The data type (string, number, or date/time)
  - The value; the following is an example duration class item:

    D=1-5 & HH=08-20

    You specify values by using regular expressions. Use the same expressions defined when "Defining Filter Criteria".

> **Note:**
>
> For values that do not match any other class items, specify an "undefined" class item for every class.

About linked classes

You can create multiple linked classes to group results by more than one category. For example, if you create groups for time period and duration, you can group results as follows:

- Peak time: 0-1 minute
- Peak time: 1-5 minutes
- Peak time: over 5 minutes

- Off-peak time: 0-1 minute
- Off-peak time: 1-5 minutes
- Off-peak time: over 5 minutes

About Defining Class Dependencies

During aggregation, class groupings are processed in order as defined in a tree structure. To define this order, you assign classes to nodes in the tree structure. For example, to aggregate data by time and duration, data is aggregated first by the time class, and then by the duration class.

8

Migrating Pipeline Manager Data between Test and Production Systems

This chapter explains the Pipeline Manager data migration features of Oracle Communications Billing and Revenue Management (BRM). It provides conceptual information about Pipeline Manager data migration and operational information about implementing data migration features in your business.

About Pipeline Manager Data Migration

You use the Pipeline Manager data migration feature to create, test, and modify pipeline rating data in a development environment, then deploy it to your production environment. This capability provides increased flexibility and security by isolating development and testing activities from production systems.

This is the basic workflow of the Pipeline Manager data migration process:

1. **Set up identical development and production systems.**

   To ensure data integrity and testing validity, the pipeline rating data in the development system must be exactly the same as the data in the production system. The two systems diverge as you make changes to the development environment, but they converge again when you export these changes and import them into the production system.

   See "Setting Up Development and Production Environments".

2. **Create and modify charges and discounts on the development system.**

   Using standard Pricing Center procedures with some modifications, you create and modify pipeline pricing data such as charges, discounts, and pricing. You use change sets to organize your work. Change sets are groups of related changes that are managed and exported as a whole. Change sets ensure data integrity by locking objects when necessary.

   See "Understanding Change Sets" and "Understanding Locks and Associations".

3. **Test your changes on the development system.**

   You should test your changes thoroughly to ensure that they work as you expect. The data migration features in Pricing Center guard against major errors such as referring to objects that don't exist, but you must test the business content of your changes to make sure that you achieve the results that you want.

   See "Testing Change Sets".

4. **Export change sets from the development system.**

   After testing, you can export a single change set or a group of change sets to a file. The file contains the information required to recreate the modified objects in the production database. If you export a group of change sets to a single file, you can specify the order in which the change sets are exported.

   See "Planning the Export Process".

5. **Import change set files into the production environment.**

The import process reads the file created during export from the development system. All changes are validated to ensure data integrity. For example, the import process checks that all objects referred to by objects in the change set exist in the production database. If errors are found, the entire file is rejected and no changes are made.

See "Planning the Import Process".

Understanding Change Sets

Change sets are the basis of the Pipeline Manager data migration features in Pricing Center. Change sets track the changes you make and make it possible to treat those changes as a whole. They also enforce rules about which objects can be added, changed, or deleted. See "Understanding Locks and Associations".

For example, a change in policies at your business makes it necessary to modify several pricings. You can make all these modifications as part of single change set that you export from the development system and import into the production system as a whole. Using a change set guarantees that the production system receives exactly the same changes that you made in the development system.

The content of a change set is determined by the changes you make while that change set is *active*. When data migration is enabled, you must have an active change set before you make any changes to rating and pricing data. You can have only one active change set at a time and a change set can be active for only one user at a time. Every change you make is part of the active change set.

You can activate, inactivate, and reactivate change sets freely. For example, you can activate one change set to make a change to a screen, then switch to another change set to make a different change. When you exit from Pricing Center, the active change set is automatically inactivated to make it available to other users.

You can create as many change sets as you need. In some cases you may use only a single change set to include all the changes associated with a particular pricing change. In other cases, you may need to set up a number of different change sets for various parts of the project. See "Organizing Work into Change Sets".

You use the Change Set Manager, a screen in the Pricing Center application, to create and manage change sets. See "About the Change Set Manager ".

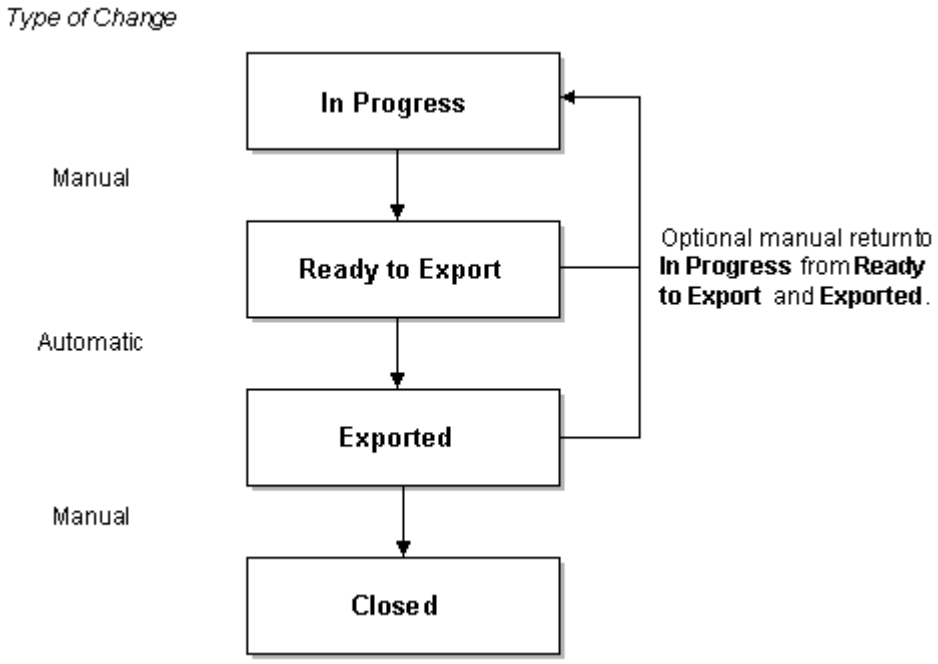
Understanding the Change Set Life Cycle

Change sets follow a life cycle that dictates what you can and can't do. This life cycle is comprised of change set states through which change sets move. By default, the life cycle includes four states. You can add additional states to conform to your business practices.

> **Note:**
>
> Managing the change set life cycle is a sensitive task. It is possible to invalidate testing scenarios and to create discrepancies between the test and production environments if you do not monitor and manage change set states carefully.

Figure 8-1 illustrates the default change set life cycle:

Figure 8-1 Default Change Set Life Cycle



There are four default change set states:

- **In Progress**. When you create a new change set, it is in the In Progress state, the working state for change sets. When an In Progress change set is active, you can make any modifications to the pipeline rating data that you need. From In Progress, you can manually change the state to Ready to Export or to a custom states that you define.
- **Ready to Export.** You change the state to Ready to Export when you confirm that all your data is complete and correct. You should test your data before switching to this state.

  In the Ready to Export state, you cannot make any changes to the data in the change set. If you need to make additional changes, you must manually switch back to In Progress.

  The normal next step is to export the change set to a file. When you export a change set, its state automatically switches to Exported.
- **Exported**. This state shows that the change set has been exported, but has not yet been imported into the production database. You cannot make any changes to the data in a change set in the Exported state.

  You must manually change the state to Closed when the file is imported.

> **Note:**
>
> Changing to Closed state from Exported is a vital step. If you do not close the change set, all locks and associations remain active, potentially blocking the completion of other change sets.

If there is a problem importing the change set, you can manually return the change set to the In Progress state to make necessary changes. This should be a carefully managed

task to avoid confusion about which change set file contains the correct data. See "Planning the Export Process".

- **Closed**. This is the end state for change sets. The Closed state indicates that a change set is complete and in production. The change set is no longer available for use and cannot be reactivated. Its locks and associations are released. You can view information about the modifications made in this change set, but can no longer back them out.

For more information, see "Customizing Change Set States".

Understanding Locks and Associations

Locks and associations are used by change sets to ensure data integrity, prevent contradictory changes, and maintain information about what data has changed or been affected.

- A *lock* prevents a data object from being modified or deleted by a change set other than the one that established the lock.
- An *association* indicates that a data object is referred to by a locked object. While an object can have only a single lock, it can have multiple associations. It can also have a lock and associations at the same time. An object with an association cannot be deleted until the association is removed.

When data migration is enabled, locks and associations are automatically implemented by Pricing Center, preventing you from making changes that violate the locking rules. For example, if a change set has locked a pricing, you can't modify that pricing with any other change set.

Even though locks and associations are enforced automatically, you should understand the rules that are used to enforce them. This knowledge will help you and your colleagues work efficiently and smoothly. For example, if you create a new calendar in a particular change set, locking prevents that calendar from being visible to other change sets until the change set that created the calendar is closed. You need to plan your work accordingly. See "Organizing Work into Change Sets" for more information.

The next section, "Understanding the Pricing Data Model", provides background information about how pipeline rating data is stored in the database. "Locking and Association Rules" provides information about the rules governing locks and associations.

Understanding the Pricing Data Model

When you create a charge, pricing, or another element of pipeline rating data, it is stored in a table in the Pipeline Manager database, but for the purposes of clarity, we can think of Pipeline Manager data as independent objects.

These objects have different kinds of relationships with each other. Many objects are *reusable*. They are elements such as calendars, time models, pricings, and zone models that can be referred to by many other objects. For example, a single calendar can be used by many charges.

Other objects have a *parent-child* relationship. One parent object refers to many different occurrences of the same type of child object.

Locking and Association Rules

When an object is modified, added, or deleted while a change set is active, that change set has a lock on the object. For example, if you make a change to a charge under Change Set 1, that charge is locked by Change Set 1 and cannot be modified or deleted by another change set until the lock is released.

These are the three most basic locking rules:

- An object can have only one lock. In other words, when an object is locked by one change set, it cannot be locked by another.
- An object that is locked by a change set cannot be modified or deleted by another change set.
- A newly created object is locked by the change set that created it.

These rules prevent change sets from making contradictory changes.

In addition to these basic locking rules, additional rules govern the objects related to locked objects. These rules work differently depending on whether the object is reusable or part of a parent-child relationship.

Rules for reusable objects

When a change set locks an object that directly refers to a reusable object, the change set creates an association to the reusable object. For example, when you modify a charge, the change set creates a lock on the charge and an association to the calendar referred to by the charge.

Associations are used to keep track of data that is potentially affected by changes made in a change set. They are also used to guard against deletion of objects that might cause data integrity problems.

These are the rules governing locking and associations for reusable objects:

- Objects with associations cannot be deleted until all associations have been removed. The deletion of associated objects is prohibited because it would cause data integrity problems; the locked object would have a reference to an object that no longer exists.
- Multiple change sets can create associations to the same object. For example, suppose Charge A and Charge B both refer to Calendar Z. Change Set 1 modifies Charge A, thereby creating an association to Calendar Z. Change Set 2 then modifies Charge B, which creates an additional association to Calendar Z.
- A change set can create an association to a locked object, except if that object is newly added by another change set. For example, if Change Set 1 has modified Pricing A and therefore locked it, Change Set 2 can still make a change that results in an association to that pricing. However, if Change Set 1 has *added* Pricing B, the new pricing is not visible to other change sets. No associations can be created to it by other change sets.
- A change set can obtain a lock on an associated object to make modifications. (The object cannot be deleted, however.) For example, suppose Change Set 1 modifies Charge A, which locks the charge and creates an association to Calendar Z. Change Set 2 can still lock the calendar for modification. It cannot delete the calendar because that would violate the rule about deleting associated objects.
- Associations are created only for objects directly referred to by a particular locked object. When you lock a charge, you create an association to the calendar it refers to, but you do not create an association to any objects referred to by the calendar.

Rules for parent-child relationships

There is one main locking rule for parent-child relationships: a child object is locked when its immediate parent object is locked. For example, when you modify a charge, the charge and all its charge versions are locked.

Unlike associations, parent-child locking is recursive. In other words, not only the children of the parent object, but the children's children, are locked. For example, when you lock a charge,

its versions are locked, which in turn causes the version's charge configurations, rate adjustments, and special day links also to be locked.

Because of the recursive nature of parent-child locks, you need to use some special techniques to minimize their impact. For example, to prevent all of a charge's children from being locked when you want to modify only a particular version, you can open the charge in read-only mode, then open the version you want to edit. Only that version and its children are locked, making it possible for other change sets to change other parts of the charge.

> **Note:**
>
> You cannot *delete* a child object when you open its parent in read-only mode. For example, if you open a charge in read-only mode, then try to delete a charge version, you see an error dialog box. You must open and lock the parent object to delete a child object.

About the Change Set Manager

You use the Change Set Manager in Pricing Center to create and work with change sets. The Change Set Manager has two main areas:

- The navigation panel on the left enables you to open and create change sets. It lists all the available change sets in two sections: **Non-Exported** and **Exported**. You also search for closed change sets in the navigation panel.
- The Change Set Manager window displays the details of the open change set, including the name, state, description., the data modified by this change set, and the data associated with it. You can also activate a change set in the Change Set Manager window.

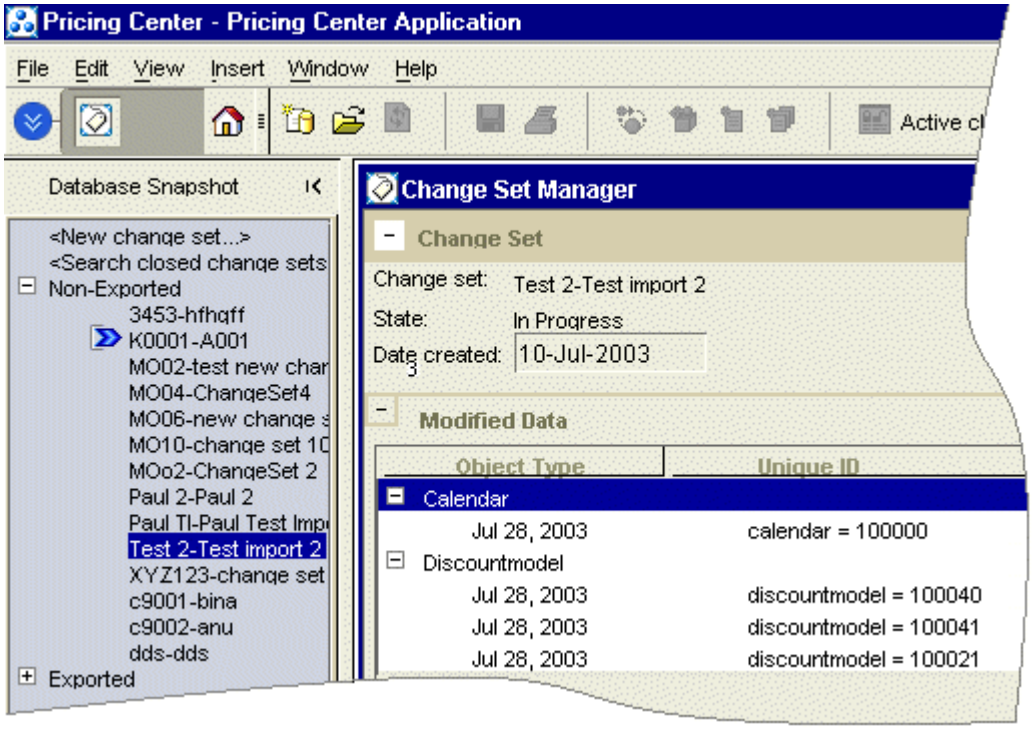
To open the Change Set Manager:

- Click the Change Set Manager button in the toolbar.

Alternatively, choose **View - Change Set Manager**.

The Change Set Manager opens. The navigation panel on the left side of the screen shows the available change sets as shown in Figure 8-2.

Figure 8-2 Change Set Manager



In the Change Set Manager, you can do the following:

- Create new change sets
- Activate change sets
- View change set data
- Work with change set states
- Back out change sets
- Export change sets
- Import change sets

Using Pipeline Manager Data Migration Features in Your Business

The Pipeline Manager data migration features in Pricing Center provide a framework that you can use to create and test new pricing data for your business. Because every business is different, you need to develop procedures that meet your needs. This section provides guidance about integrating Pricing Center data migration features into your business.

> **Note:**
>
> The Pricing Center data migration features guard against problems in data integrity, but don't validate content. You must be careful and methodical to ensure that your business content is correct and testable.

Setting Up Development and Production Environments

To ensure accurate testing, your development and production data models must be identical to begin with. Each database must contain exactly the same objects with exactly the same contents.

When you create new pricing data on the development system, the two databases diverge, but due to the data that you have introduced. You can therefore be confident that your testing will reveal only issues introduced by the new data. Later, when all your changes have been exported from your development system to your production system, the data models will again be identical. See "Copying Production Data to the Development System".

You must also enable data migration for the two systems. When you install Pricing Center, you can choose not to enable data migration, to enable only import capability, or to enable the entire feature. You can also enable or disable data migration after Pricing Center has been installed. See "Enabling Data Migration in Pricing Center".

Until you enable data migration, you cannot import or export data to or from either system.

> **Note:**
>
> If you enable data migration for one instance of Pricing Center, you should also enable data migration for *all* other instances of Pricing Center that have access to the same test and production databases. Instances of Pricing Center without data migration enabled can make changes outside the scope of change sets, thereby causing inconsistencies in the data.

Another configuration step is setting up the change set life cycle that you want to use. The change set life cycle is based on your business process and reflects the stages that a change goes through from development to production. Depending on your business process, you may want to add change set states to the default life cycle. For example, you may want to add states called Testing and Approval. See "Understanding the Change Set Life Cycle" and "Customizing Change Set States".

Planning Your Work

You should plan your pricing data projects offline before using Pricing Center to implement them on the development system. You should know exactly which new pricing objects you need to introduce, which objects need to be changed, and which need to be deleted. You should also make sure to identify all reusable object that will be modified so that you can test all the objects that refer to them.

The exact planning process you use depends on your business needs and the complexity of the data model you are implementing. Whatever process you use, make sure there is a solid connection between your offline process and your work in Pricing Center. Use a consistent

naming policy so that you can track a change from its inception in the planning process to its implementation in Pricing Center. For example, if your marketing department initiates a change through a formal change request, you can incorporate information from the change request into the Change Set ID, name, and description in Pricing Center.

Organizing Work into Change Sets

The way you organize change sets on your development system; how many change sets you use for a particular project and how changes are divided among them; can have a large impact on how efficiently you complete your work. Here are two reasons why you need to think carefully about change set organization:

- Change sets can be dependent on each other. For example, if you are creating a new reusable object in one change set, other change sets cannot view or reference that object until the first change set is closed. Dependencies can also be based on content. If you make changes to the content of a pricing in one change set, for example, any changes you make in other change sets that rely on this new content create a dependency.
- Locks created by one change set can prevent other change sets from accessing objects. For example, suppose two people are implementing two separate groups of changes, each in its own change set, and each change set requires the modification of the same pricing. When one user modifies the pricing, the other user is blocked until the first change set is closed.

You should use the information from the planning process to decide how many change sets to create and what to include in them. From your planning, you should develop a clear picture of the specific changes you need to make.

These are some of the factors to consider when organizing change sets:

- **The number and complexity of changes.** If you need to make only a few, simple changes, you can make them all in single change set. On the other, if you are working on a major overhaul of your pricing data, you should organize your work into multiple change sets.
- **The types of changes you are making.** Reusable objects can be referenced by many different objects, so changes to them can have a broad impact. To prevent one change set's locks from causing problems for other change sets, you can make all your reusable object changes in one change set that you export and import separately before other changes.
- **How many users are involved.** The larger the number of users involved in the implementation of a pricing data project, the more important it is to be very careful about planning and organizing the work. You can use change sets to divide work among users.

Testing Change Sets

Testing is vital to ensure that the pricing data you create is valid. Pricing Center prevents you from making errors that cause data integrity problems such as references to objects that no longer exist. But it is your responsibility to ensure that the *content* of your pricing data is valid: Pricing Center can ensure that a pricing exists, but it can't verify that it contains the correct information for your business.

Pricing Center does provide warnings in situations where your actions might cause problems. For example, when you open a Pricing Center screen for an object that has an association to another change set, you can get information about which change set created the association.

You should test the data as realistically as possible by using the pricing data in your development system to rate CDRs in an environment that closely resembles your production environment.

These are some guidelines for helping you decide what to test:

- When you modify a reusable object, test to ensure that the objects that reference it are still valid. For example, if you make a change to a calendar, you should test all the charges that refer to that calendar to make sure that the change doesn't cause an unexpected result.
- Keep in mind the possible effects of multiple successive change sets modifying the same object. Locking prevents more than one change set from modifying an object at the same time. After a change set is closed and its locks are released, however, another change set can modify previously locked objects.
- Coordinate the activities of all users and all change sets to ensure that you are testing exactly what will be exported and imported. For example, another change set can modify a pricing to which a charge in your change set has an association. Such a change doesn't cause a referential integrity problem; the pricing still exists; but may cause unexpected results during rating. Ideally, when you are testing a change set or group of change sets, there is no other development activity that might affect the tests.

Planning the Export Process

When a change set or group of change sets is complete, you export it to a file that can be imported into the target system.

Follow these guidelines for exporting change sets:

- Before you export, make sure that you understand any dependencies between change sets.

  Some dependencies are determined by locking rules. For example, if a new reusable object is introduced in one change set, that change set must be exported, imported, and closed before another change set can refer to the new object.

  Other dependencies are based on content. For example, if you use a change set to modify an existing calendar, you should make sure to export and import that change set before any change sets that rely on the modification.
- To ensure that change sets are exported and imported in the proper order, you can include multiple change sets in the same file and specify the sequence in which they are processed.
- Export change sets only when you are sure that they are complete and ready to be imported. There is no point in having incomplete export files in your system. They serve no purpose and there is some risk that they might be imported accidentally.

Managing Change Set Files

Change set files contain sensitive data, so you should manage them carefully. Three tasks are particularly vital:

- **Ensuring file security.** Once a file has been exported, it should be tracked carefully to ensure that it is not lost or corrupted. You should make sure there is no confusion about file names, which files are waiting to be imported, and similar matters.
- **Keeping track of file versions.** It's possible to have more than one version of a change set file. For example, if you export a file and then discover a problem with a change set in the file, you may need to make corrections and export the changes again. You should be

very careful to keep track of the file versions to ensure that you are importing the correct one.

- **Keeping track of the file sequence.** In some cases, files must be exported and imported in a particular order. For example, if you modify a calendar in one change set, you should export and import that change set before other change sets that rely on the modified calendar.

Depending on the complexity of your data model, a version control system may be useful for managing change set files.

You can specify the default locations to which change set files are exported and from which they are imported. For example, you can choose to export files to a location known to your version control system.

Planning the Import Process

Importing data into your production system is obviously a critical task. The Pricing Center import process checks every change to ensure that it is valid. If there are any errors during importation, the entire file is rejected, no changes are made to the target data, and the file is moved to an error directory.

In addition, you can take these steps to ensure that data is imported successfully:

- **Import files in the correct order.** If there are content dependencies among change set files, make sure to import them in the required order. For example, a change set may include a modification that another change relies on. The locking rules and other safeguards prevent data integrity errors such as references to non-existent objects, but you must keep track of dependencies based on business content.
- **Ensure that files reflect final data.** You should check that you are importing files that contain the final versions of your pricing data. If you accidentally import a file that is incomplete, you have to remove or modify the data manually; importing a file cannot be reversed. Careful planning and file management will prevent these problems.

Coordinating Real-Time Rating Data Migration and Pipeline Data Migration

BRM packages contain a mixture of real-time and pipeline data. For example, when you define charge offers, you can map some events to real-time charges and other events to pipeline charges.

Real-time and pipeline pricing data are exported and imported separately using different procedures. This document covers data migration for pipeline pricing data only.

You must manually coordinate the real-time and pipeline migration procedures. For example, if you added new pipeline pricing data associated with rating a new charge offer, you must migrate both the new charge offer (real-time data) and the new pricing data (pipeline data).

Configuring Pipeline Manager Data Migration Features

There are a number of configuration task for Pipeline Manager data migration that you accomplish outside the Pricing Center application, including enabling data migration, copying data to ensure that the development and production systems are identical, and optionally customizing the change set life cycle.

Enabling Data Migration in Pricing Center

The Pipeline Manager data migration features are optional. They are visible only if you enable them. You can enable the ability to import change set files without enabling the full set of data migration features. Import-only data migration is useful for production systems where you want to reduce the risk of accidental data changes.

You normally enable data migration during the Pricing Center installation process. You can also enable data migration after Pricing Center has been installed by making a change to the Pricing Center configuration file.

> **Note:**
>
> If you enable data migration for one instance of Pricing Center, you should also enable data migration for all other instances of Pricing Center that have access to the same test and production databases. Instances of Pricing Center without data migration enabled can make changes outside the scope of change sets, thereby causing inconsistencies in the data.

To enable data migration after installation:

1. Exit Pricing Center.
2. Open the Pricing Center configuration file (**custom.properties**) in a text editor. This file is located in the **\lib** subdirectory of the installation directory, normally **C:\Program Files\Portal\PricingCenter**.
3. Change the value for the **pipeline.datamigration** parameter to **2** (for full data migration functionality) or **1** (for import capability only).
4. Save the file.
5. Start Pricing Center.

Copying Production Data to the Development System

When you set up your development environment, you start out with an exact duplicate of the production database.

> **Note:**
>
> The test and production databases must be completely identical, including sequence IDs, for data migration to work reliably.

Use the replication tools provided with your database to copy the production database. See your system's documentation for instructions.

Customizing Change Set States

You can customize change set states to define a workflow for your projects. Your need for this customization depends on the complexity of your work. If there are only one or two change

sets in progress at any one time and they contain simple changes, it may not be necessary to customize. On the other hand, if you have a team of planners working on multiple change sets in varying states of completion, you should customize the life cycle to reflect your process. See "Understanding the Change Set Life Cycle".

To customize change set states, you modify the **state.config** file, then load it by using the **stateconfigtool** utility. The **state.config** file lists each valid state transition. In other words, it defines all the states that it is valid to move to from any given state. It also lists whether the transition is manual (accomplished by the user in the Change Set State dialog box) or automatic (accomplished by the BRM).

These are the contents of the default **state.config** file, which defines the standard change set life cycle:

```
# State Transition for Changeset
# currentState,nextState,Action

inprogress,readytoexport,manual
readytoexport,exported,automatic
readytoexport,inprogress,manual
exported,inprogress,manual
exported,closed,manual
```

> **Note:**
>
> You can define additional states, but you cannot delete any default states. The custom states you define must come between In Progress and Ready to Export. All customized states must have a manual transition.

For example, the following file defines a new state called Testing. You can switch to Testing from In Progress and can switch from Testing to Ready to Export or back to In Progress. Note that you can't switch from Ready to Export back to Testing.

```
# State Transition for Changeset
# currentState,nextState,Action

inprogress,readytoexport,manual
inprogress,testing,manual
testing,readytoexport,manual
testing,inprogress,manual
readytoexport,exported,automatic
readytoexport,inprogress,manual
exported,inprogress,manual
exported,closed,manual
```

You load the change set configuration into the database by using the **stateconfigtool** utility. When you run this utility, you specify the file name, the database type, the host, the port number, the database instance, a login user name, and a login password.

To customize change set states:

1. Use a text editor to open the **state.config** file located in the Pricing Center directory, typically **Program Files\Portal Software\Pricing Center**.

2. Add new states, being careful to account for all the transitions necessary.

> **Note:**
>
> Don't make any changes to the default entries in the **state.config** file. Doing so will cause an error when you load the file.

3. Save the file under a new name. Saving the file under a different name preserves the original file in case you want to return to the default configuration.
4. From the *pipeline_home***tools\StateConfigTool** directory, run the **stateconfigtool** utility to load the file. *pipeline_home* is the directory where you installed Pipeline Manager. Use this syntax:

```
stateconfigtool -f file_name -d database_type -h host -n port -i database_id
```

For example:

```
stateconfigtool -f pipeline_home/tools/StateConfigTool/state.config -d oracle -h
sample_host -n 1521 -i pindb
```

Exporting and Importing Change Sets by Using the loadchangesets Utility

Under certain circumstances, importing and exporting change sets by using Pricing Center may be inconvenient or undesirable. For example, you may prefer not to allow Pricing Center access to your production system to prevent unauthorized or accidental changes to your production pricing data.

In these cases, you can use the **loadchangesets** command-line utility to export change sets from your test environment and import them into your production database.

> **Note:**
>
> Exporting and importing change sets by using the **loadchangesets** utility changes only the final stages of the entire pipeline pricing data migration process. You still use Pricing Center to create and manage change sets.

The general process for exporting and importing change sets is similar whether you use Pricing Center or **loadchangesets**.

The **loadchangesets** utility has two modes: interactive and non-interactive. They both enable you to import and export change sets, but work somewhat differently. See "Working in Interactive and Non-Interactive Mode".

Unlike Pricing Center, where you can choose which change sets to export, **loadchangesets** exports all the change sets that are in Ready to Export state. You should therefore be careful to monitor the life cycles of your change sets to ensure that you are exporting all the changes sets you want and none that you don't want.

Specifying BRM Servers for the loadchangesets Utility

Before you can use the **loadchangesets** utility, you must specify the BRM servers to export from and import to. You enter the host names (or IP addresses) and port numbers in a configuration file.

To specify BRM servers for import and export:

1. Exit Pricing Center if necessary.
2. Open the Pricing Center configuration file (**custom.properties**) in a text editor. This file is located in the **\lib** subdirectory of the installation directory, normally **C:\Program Files\Portal\PricingCenter**.
3. To specify the server from which to export pricing data, enter the host name (or IP address) and port number in the **pipeline.datamigration.utility.export.environment.host** and **pipeline.datamigration.utility.export.environment.port** entries.

   For example, to export from TestPricingServer, port number 11960, enter the following:

```
pipeline.datamigration.utility.export.environment.host=TestPricingServer
pipeline.datamigration.utility.export.environment.port=11960
```

4. To specify the server to which to import pricing data, enter the host name (or IP address) and port number in the **pipeline.datamigration.utility.import.environment.host** and **pipeline.datamigration.utility.import.environment.port** entries.

   For example, to import to ProductionPricingServer, port number 56968, enter the following:

```
pipeline.datamigration.utility.import.environment.host=ProductionPricingServer
pipeline.datamigration.utility.import.environment.port=56968
```

5. Save the file.

Working in Interactive and Non-Interactive Mode

You can use the **loadchangesets** utility in interactive or non-interactive mode:

- In interactive mode, you issue a command for each step in the process of importing or exporting. After you enter interactive mode, the prompt changes to an angle bracket and commands are single words for performing particular actions. You can view a list of the change sets that will be imported and exported
- In non-interactive mode, you use commands that batch several related parts of the import or export process. You must enter a full command, including the utility name for each set of actions.

Exporting and Importing Change Sets in Interactive Mode

The following procedures describe exporting and importing change sets by using **loadchangesets** in interactive mode.

To export and import change sets by using **loadchangesets** in interactive mode:

1. Make sure the change sets that you want to export are complete and that they, and no others, are in Ready to Export state.
2. Go the **/lib** directory in the Pricing Center installation directory.

3. To switch to interactive mode, enter the following command:

   ```
   loadchangesets -i
   ```

   The prompt changes to **==>.**

4. To load the change set data from the export database into memory, enter the following command:

   ```
   export
   ```

5. To write the change set data from memory to a change set file, enter the following command. The file name must include the .**exp** file name extension.

   write *change_set_file*

   The change sets are exported to the specified file in the **/export/done** subdirectory in the Pricing Center installation directory. This directory is created automatically if it doesn't exist when you run the utility.

6. Move the change set file that you created into the **/import** subdirectory in the PricingCenter install directory.

7. To read the change set data from the change set file into memory, enter the following command. The file name must include the **.exp** extension.

   read *change_set_file*

8. (Optional) Enter the following command to view the names of the change sets you loaded into memory:

   ```
   list
   ```

9. To load the change set data from memory into the import database, enter the following command:

   ```
   import
   ```

Exporting and Importing Change Sets in Non-Interactive Mode

The following procedures describe exporting and importing change sets by using **loadchangesets** in non-interactive mode.

To export change sets by using **loadchangesets** in non-interactive mode:

1. Make sure the change sets that you want to export are complete and that they, and no others, are in Ready to Export state.
2. Go the **/lib** directory in the Pricing Center installation directory.
3. Enter the following command, where *change_set_file* is the name of the file into which you want to export the change sets. The file name must include the .**exp** file name extension.

   loadchangesets -fx *change_set_file*

The change sets are exported to the specified file in the **/export/done** subdirectory in the Pricing Center installation directory. This directory is created automatically if it doesn't exist when you run the utility.

To import change sets by using **loadchangesets** in non-interactive mode:

1. Move the change set file that you want to import into the **/import** subdirectory in the PricingCenter install directory.
2. Go the **/lib** directory in the Pricing Center installation directory.
3. Enter the following command, where *change_set_file* is the name of the file that you want to import. The file name must include the **.exp** file name extension.

   ```
   loadchangesets -fi change_set_file
   ```

9

Transferring Data Between Pipeline Manager Databases

This chapter describes how to transfer data from one Pipeline Manager database to another, such as from a test database to a production database, by using the Oracle Communications Billing and Revenue Management (BRM) **LoadIfwConfig** utility.

About Transferring Data

You transfer data by extracting data from a *source* Pipeline Manager database and then loading the data into a *destination* Pipeline Manager database. You specify which data to extract at the command line or by using an XML file. The **LoadIfwConfig** utility extracts the specified data from the source database to an output XML file. The utility can then load the output XML file directly into the destination database.

About Specifying the Data to Extract

You can specify to extract:

- All data in the Pipeline Manager database. You specify to extract all data by using only a utility command. For instructions, see "Extracting All Database Objects with LoadIfwConfig".
- All Pipeline Manager data that has been modified after a specified date and time. You specify to extract data based on the modification date and time by using utility commands. For instructions, see "Extracting All Database Objects Modified after a Specific Time".
- A subset of the data, based on the objects' attributes and modification time. You specify to extract a subset of the data by using an input XML file with the **LoadIfwConfig** utility. See "About Creating an Input XML File to Extract Data".

About Creating an Input XML File to Extract Data

If you are extracting a subset of data, you must create an input XML file that specifies the table from which to extract the data and, optionally, the criteria that the data must meet. The criteria consists of fields and their required values. For example, you can specify to extract from a specific table only those objects that have the SAMPLE field set to 100. The utility would then extract all objects with a SAMPLE field set to 100 as well as any child objects and any dependent objects. See "About Specifying to Extract Child and Dependent Objects".

When the input XML file specifies a table only, the utility extracts objects from the entire table as well as from any child and dependent objects. When the input XML file specifies a table and required field values, the utility extracts from the table only those objects that contain the matching field values plus any child and dependent objects.

The syntax for the input XML file is shown below:

```
<RecordSet>
  <TableName [FieldName1 ="Value1"] [FieldName2 ="Value2"] .../>
</RecordSet>
```

where:

- *TableName* is the name of the table from which to extract objects.
- *FieldNameX* is the name of the table field that must match the specified value. You can list multiple field-value pairs.
- *ValueX* specifies the required field value. To be able to list multiple values for each field, see "About Using Regular Expressions when Specifying the Data to Extract".

For example, the following input XML file specifies to retrieve all objects from the IFW_RUM table that match all of the criteria below:

- Have their NAME field set to **Duration**.
- Have their RUM field set to **DUR**.
- Have their TYPE field set to **D**.

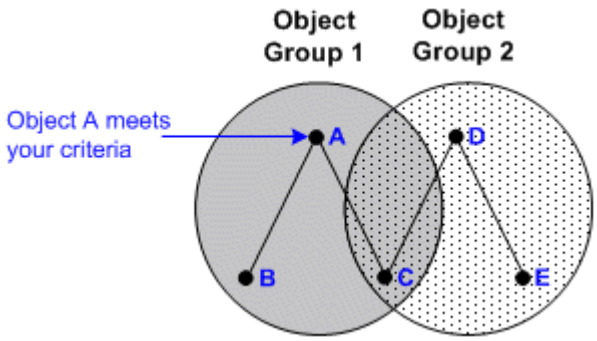
```
<RecordSet>
   <IFW_RUM NAME ="Duration" RUM ="DUR" TYPE ="D" />
</RecordSet>
```

About Specifying to Extract Child and Dependent Objects

When extracting a subset of data, the **LoadIfwConfig** utility automatically extracts all objects that meet your specified criteria as well as any related child objects. For example, when the utility extracts a charge object, the utility automatically extracts all child objects of the charge, such as the charge version and charge configuration objects.

If one of the child objects is also related to other objects, you can configure the utility to extract the other objects as well.

Figure 9-1 Object Relationships and Dependencies



For example, in Figure 9-1 above, object A meets your specified criteria. Because object A is a parent object, the utility automatically extracts object A and its two child objects, B and C. Because object C is also related to object D, extracting only the objects in object group 1 would break the relationships in object group 2.

To prevent this from happening, you can configure the utility to extract object D whenever object C is extracted. You do this by making object D a dependent of object C. In this case, whenever object C is extracted, the utility would also extract:

- Object D, because it is a dependent of object C.
- Object E, because it is a child of object D.

The utility determines the object dependencies by using the *pipeline_home***/tools/XmlLoader/LoadIfwConfig.xsd** file, where *pipeline_home* is the directory where you installed Pipeline Manager. You customize the dependencies by using the *pipeline_home***/tools/XmlLoader/CustomConfig.xml** file.

> **Note:**
>
> The settings in the **CustomConfig.xml** file override the settings in the **LoadIfwConfig.xsd** file.

The syntax for the **CustomConfig.xml** file is shown below:

```
<TableName AddDependingTable="AddTable" AddDependingTableMapping="Field1:Field2"/>
<TableName DependingTableNames="DependingTable"/>
```

where:

- *TableName* specifies the table that has dependent objects in another table.
- *AddTable* specifies the dependent table. The utility extracts data using the provided mapping whenever data from *TableName* is extracted.
- *Field1* is the field from *TableName* that is related to a field in *AddTable*.
- *Field2* is the relating field in *AddTable*. The utility extracts any objects from *AddTable* that have matching values whenever data from *TableName* is extracted.
- *DependingTable* specifies the list of dependent tables. Data from these tables must be extracted whenever data is extracted from *TableName*. You can list multiple table names by using the pipe symbol (**|**) as a delimiter.

Sample **CustomConfig.xml** entries are shown below:

```
<IFW_RATEPLAN_CNF AddDependingTable="IFW_TIMEMODEL" AddDependingTableMapping="TIMEMODEL:TIMEMODEL"/>
<IFW_RATEPLAN_CNF DependingTableNames="IFW_TIMEMODEL|IFW_PRICEMODEL"/>
```

- The first line specifies that the utility must extract dependent data from IFW_TIMEMODEL, relating the TIMEMODEL field of IFW_RATEPLAN_CNF to the TIMEMODEL field of IFW_TIMEMODEL.
- The second line specifies to extract dependent data from the IFW_TIMEMODEL and IFW_PRICEMODEL tables whenever data is extracted from IFW_RATEPLAN_CNF.

About Using Regular Expressions when Specifying the Data to Extract

By default, you cannot use regular expressions when specifying the data to extract. This means that the input XML file must include a separate line for each required field value, which impacts performance because multiple entries generate multiple SQL queries to the database. For example, to retrieve objects from the IFW_RATEPLAN_CNF table that have an IMPACT_CATEGORY value of FRANCE or SPAIN, the input XML file would contain these lines:

```
<IFW_RATEPLAN_CNF IMPACT_CATEGORY="FRANCE"/>
<IFW_RATEPLAN_CNF IMPACT_CATEGORY="SPAIN"/>
```

You can configure the utility to accept the following regular expressions when searching defined fields:

- The asterisk (*****) symbol for wildcard searches.

- The pipe (**|**) symbol for the logical OR operation.

You define which fields support regular expressions by configuring the **RegExFields** entry in the *pipeline_home***/tools/XmlLoader/CustomConfig.xml** file.

The syntax for the **RegExFields** entry is shown below:

```
<TableName RegExFields="FieldName"/>
```

where:

- *TableName* is the name of the table that contains the specified field.
- *FieldName* is the name of the field that can be searched with regular expressions. You can list multiple fields by using the pipe symbol (**|**) as a delimiter.

For example, the following entry specifies that you can use regular expressions when searching for values in the IMPACT_CATEGORY field of the IFW_RATEPLAN_CNF table:

```
<IFW_RATEPLAN_CNF RegExFields="IMPACT_CATEGORY"/>
```

For the above example, you could retrieve records that have their IMPACT_CATEGORY field set to FRANCE or SPAIN by using this input XML entry:

```
<IFW_RATEPLAN_CNF IMPACT_CATEGORY="FRANCE|SPAIN"/>
```

About the LoadIfwConfig Error Messages

The **LoadIfwConfig** utility logs information about any errors it encounters to the log file you specified in the **ProcessLog** section of the **LoadIfwConfig.reg** registry file. Table 9-1 describes the utility's error messages.

> **Note:**
>
> The utility will pass through any error messages thrown by the Xerces SAX parser and the Oracle database. For information about these error messages, see the appropriate vendor's documentation.

Table 9-1 LoadIfwConfig Error Messages utility

Error message	Description
ERROR: Connection is not Valid	The utility failed to connect to the database.
ERROR: DataBaseStatus is not Valid	The database credentials or database connect string is not correct.
ERROR: Couldn't get next sequence: *SQLString*	The utility could not generate a sequence number to insert into the database.
ERROR: during insert: *SQLString*	The utility encountered an error during the insert operation.
ERROR: during update: *SQLString*	The utility encountered an error during the update operation.
ERROR: during delete: *SQLString*	The utility encountered an error during the delete operation.
Exception occurred while executing *SQLString*	The utility encountered an error while running other SQL statements.

Table 9-1 (Cont.) LoadIfwConfig Error Messages utility

Error message	Description
Exception from DB: *ErrorMessage*	The error message that was thrown by the Oracle database.
File Not parsed properly or Braces not matched properly	The utility's registry file (**LoadIfwConfig.reg**) contains incorrect entries or the entries are not framed correctly.
DependentFields structure provided in XML for depending table not proper	The **AddDependingTableMapping** entry in the **CustomConfig.xml** file is set incorrectly.
No rows for deletion because rows present in Child table	The utility could not delete the requested row because the row's associated child records still contain data.
Could not find valid Translation for: Table: *TableName* Referred Table: *TableName* Field: *FieldName* CODE Field Value: *FieldValue*	The required table dependencies are not provided in the input XML file.
FATAL ERROR at file: *FileName* line: *LineNumber* char: *Position* Message: *Message*	The input XML file contains mistakes, such as unparseable characters.

Using LoadIfwConfig to Transfer Data between Databases

To transfer data from one Pipeline Manager database to another, perform these steps:

1. Connect **LoadIfwConfig** to the source Pipeline Manager database. See "Connecting LoadIfwConfig to the Pipeline Manager Database".
2. (Optional) Specify the regular expressions and table dependencies that are supported on the source Pipeline Manager system. See "Customizing the Regular Expression and Dependent Table Settings".
3. Extract data from the source Pipeline Manager database. See "Extracting Data from a Pipeline Manager Database".
4. Connect **LoadIfwConfig** to the destination Pipeline Manager database. See "Connecting LoadIfwConfig to the Pipeline Manager Database".
5. Load data into the destination Pipeline Manager database. See "Loading Data into Pipeline Manager Databases".

Connecting LoadIfwConfig to the Pipeline Manager Database

You connect the **LoadIfwConfig** utility to the Pipeline Manager database by using the **LoadIfwConfig.reg** registry file. You can edit this registry file manually, but it is also updated by the **pin_setup** utility during the **LoadIfwConfig** installation process.

> **Note:**
>
> If you *upgraded* to Pipeline Manager 7.4, you must create the *pipeline_home***/tools/XmlLoader/log** directory before you start the **LoadIfwConfig** utility.

To connect **LoadIfwConfig** to the Pipeline Manager database:

1. Open the *pipeline_home*/**tools/XmlLoader/LoadIfwConfig.reg registry file in a text editor.**
2. **Edit the registry entries to match your system environment. In particular, pay attention to these entries:**
   - In the **XMLCustomizationFile** entry, specify the location of the optional customization XML file. **LoadIfwConfig** contains a sample customization file (**CustomConfig.xml**) and a schema file (**CustomConfig.xsd**) to which the XML should conform.
   - In the **LoadDataFromDB** entry, specify whether to increase performance by loading the sequence-to-code translation information into memory. If this entry is missing or blank, it defaults to **False**.
   - (Optional) In the **RowFetchSize** entry, specify the number of database rows to retrieve on each trip to the database. Increasing the number of rows can reduce the required number of database fetches and increase the utility's performance. The default is **100**.

   The other entries are standard logging and connection registry entries.
3. Save and close the file.

A sample **LoadIfwConfig.reg** file is shown below:

```
LoadIfwConfig
{
  LogMessageTable
  {
    MessageFilePath   = ./
    MessageFileSuffix = .msg
  }

  ProcessLog
  {
    FilePath          = ./log
    FileName          = LoadIfwConfig
    FileSuffix        = .log
    WriteMessageKey   = True
  }

  DataPool
  {
    Database
    {
      ModuleName = DbInterface
      Module
      {
        # Common
        DatabaseName = DatabaseName
        UserName     = UserName
        PassWord     = EncryptedPassword
        AccessLib    = oci11g72
      }
    }
  }
  XMLCustomizationFile = CustomConfig.xml
  LoadDataFromDB = True
  RowFetchSize = 200
}
```

Customizing the Regular Expression and Dependent Table Settings

If you want the **LoadIfwConfig** utility to support regular expressions for any fields, or if you want to add table dependencies for any objects, you must modify the **CustomConfig.xml** file.

To customize the regular expression and dependent table settings, perform these steps on the source Pipeline Manager:

1. Open the *pipeline_home*/**tools/XmlLoader/CustomConfig.xml** file in a text editor.
2. Specify the table fields that support regular expressions by using the **RegExFields** entry. For example, the following entry specifies that the CODE and NAME fields in the IFW_RATEPLAN table support regular expressions:

   ```
   <IFW_RATEPLAN RegExFields="CODE|NAME"/>
   ```

   For more information, see "About Using Regular Expressions when Specifying the Data to Extract".
3. Specify any object dependencies by using the **AddDependingTable** and **AddDependingTableMapping** entries. For example, the following entry specifies to retrieve dependent data from IFW_TIMEMODEL, relating the TIMEMODEL field of IFW_RATEPLAN_CNF to the TIMEMODEL field of IFW_TIMEMODEL:

   ```
   <IFW_RATEPLAN_CNF AddDependingTable="IFW_TIMEMODEL"
   AddDependingTableMapping="TIMEMODEL:TIMEMODEL"/>
   ```

   For more information, see "About Specifying to Extract Child and Dependent Objects".
4. Specify any table dependencies by using the **DependingTableNames** entry. For example, the following entry specifies to extract dependent data from the IFW_TIMEMODEL table whenever data from IFW_RATEPLAN_CNF is extracted:

   ```
   <IFW_RATEPLAN_CNF DependingTableNames="IFW_TIMEMODEL"/>
   ```

   For more information, see "About Specifying to Extract Child and Dependent Objects".
5. Save and close the file.

Extracting Data from a Pipeline Manager Database

You can use the **LoadIfwConfig** utility to extract:

- All Pipeline Manager database objects. See "Extracting All Database Objects with LoadIfwConfig".
- All Pipeline Manager database objects modified after a certain date and time. See "Extracting All Database Objects Modified after a Specific Time".
- A subset of Pipeline Manager database objects. See "Extracting a Subset of Database Objects with LoadIfwConfig".

Extracting All Database Objects with LoadIfwConfig

The utility extracts all database objects by iterating through each table in the schema in order, selecting all of the rows, and dumping them directly into an XML file.

To extract all Pipeline Manager database objects:

1. Go to the *pipeline_home*/**tools/XmlLoader** directory.
2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   write [OutputFile]
   retrieve_all
   ```

   where *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.

   Non-interactive mode

   ```
   LoadIfwConfig  -rall [-o OutputFile]
   ```

   where *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.

   > **Note:**
   >
   > For more information about the utility's syntax, see "LoadIfwConfig".

The utility writes the extracted objects to the output file in XML format. You can now load the output XML file directly into the destination Pipeline Manager database.

Extracting All Database Objects Modified after a Specific Time

To extract all Pipeline Manager database objects that have been modified after a specified date and time:

1. Go to the *pipeline_home*/**tools/XmlLoader** directory.
2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   write [OutputFile]
   retrieve_all [-t Modifidate]
   ```

   where:

   - *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.
   - *Modifidate* specifies the timestamp after which to retrieve pricing objects. Enter the timestamp in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD*, with the server time zone as the default. For example:

     ```
     1997-07-16T19:20:30
     ```

   Non-interactive mode

```
LoadIfwConfig  -rall [-t Modifidate] [-o OutputFile]
```

where:

- *Modifidate* specifies the timestamp after which to retrieve pricing objects. Enter the timestamp in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD*, with the server time zone as the default. For example:

  ```
  1997-07-16T19:20:30
  ```

- *OutputFile* specifies the name and location of the file to which to extract the pipeline data. By default, the utility writes the output to a file named **default.out** in the current directory.

> **Note:**
>
> For more information about the utility's syntax, see "LoadIfwConfig".

The utility writes the extracted objects to the output file in XML format. You can now load the output XML file directly into the destination Pipeline Manager database.

Extracting a Subset of Database Objects with LoadIfwConfig

To extract a subset of Pipeline Manager database objects:

1. Create an XML file that lists the objects to extract. The file specifies the table from which to extract the objects and, optionally, the criteria that the objects must meet. You can use the sample input XML file (*pipeline_home*/**tools/XmlLoader/samples.xml**) as a starting point.

   See "About Specifying the Data to Extract" for more information.

2. Go to the *pipeline_home*/**tools/XmlLoader** directory.

3. Enter the following command:

   Interactive mode

   ```
   LoadIfwConfig
   [-nodep]
   read InputFile
   fetch [-t Modifidate]
   write [OutputFile]
   ```

   where:

   - **-nodep** suppresses the object dependency relationships you configured in the *pipeline_home*/**tools/XmlLoader/CustomConfig.xml** file. The utility extracts only the objects specified in *InputFile* and ignores all dependent objects.

     > **Note:**
     >
     > To suppress object dependency relationships in interactive mode, the utility session must start with the **-nodep** parameter.

   - *InputFile* specifies the name and location of the file that lists which objects to retrieve. This is the file that you created in step 1.

- *Modifidate* specifies to retrieve objects that were modified after the specified timestamp. Enter the timestamp in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD*, with the server time zone as the default. For example:

  ```
  1997-07-16T19:20:30
  ```

- *OutputFile* specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named **default.out** in the current directory.

Non-interactive mode

```
LoadIfwConfig  -i InputFile  -r [-nodep] [-t Modifidate] [-o OutputFile]
```

where:

- *InputFile* specifies the name and location of the file that lists which objects to retrieve. This is the file that you created in step 1.
- **-nodep** suppresses the object dependency relationships you configured in the *pipeline_home*/**tools/XmlLoader/CustomConfig.xml** file. The utility extracts only the objects specified in *InputFile* and ignores all dependent objects.
- *Modifidate* specifies to retrieve objects that were modified after the specified timestamp. Enter the timestamp in the ISO-8601 format: *YYYY-MM-DDThh*:*mm*:*ss* or *YYYY-MM-DD*, with the server time zone as the default. For example,

  ```
  1997-07-16T19:20:30
  ```

- *OutputFile* specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named default.out in the current directory.

The utility writes the extracted objects to the output file in XML format. You can now load the output XML file directly into the destination Pipeline Manager database.

Loading Data into Pipeline Manager Databases

You load data into the destination Pipeline Manager database by using the **LoadIfwConfig** utility's update option or insert option.

- Update option: The utility verifies whether the data provided in the input XML file already exists in the database. If the data already exists, the utility updates the database record with the new information. If the data does not exist, the utility inserts the data into the database. To update data see, "Updating the Pipeline Manager Database".
- Insert option: The utility inserts data into the database without verifying whether the data already exists. To insert data, see "Inserting Data into the Pipeline Manager Database".

Updating the Pipeline Manager Database

To update the data in the Pipeline Manager database:

1. Go to the *pipeline_home*/**tools/XmlLoader** directory.
2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   read InputFile
   update
   commit
   ```

where *InputFile* specifies the name and location of the XML file that contains the extracted pipeline data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

Non-interactive mode

```
LoadIfwConfig  -u  -c  -i InputFile
```

where *InputFile* specifies the name and location of the XML file that contains the extracted pipeline data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

> **Note:**
>
> For more information about the utility's syntax, see "LoadIfwConfig".

The utility loads the data from the XML file into the Pipeline Manager database and commits the data. If there is a failure, the utility rolls back the data and displays an error message.

Inserting Data into the Pipeline Manager Database

To insert data into a Pipeline Manager database:

1. Go to the *pipeline_home*/**tools/XmlLoader** directory.
2. Enter this command:

   Interactive mode

   ```
   LoadIfwConfig
   read InputFile
   insert
   commit
   ```

   where *InputFile* specifies the name and location of the XML file that contains the extracted data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

   Non-interactive mode

   ```
   LoadIfwConfig   -I  -c  -i InputFile
   ```

   where *InputFile* specifies the name and location of the XML file that contains the extracted data from the source database. This is the output file you generated in "Extracting Data from a Pipeline Manager Database".

   > **Note:**
   >
   > For more information about the utility's syntax, see "LoadIfwConfig".

The utility loads the data from the XML file into the Pipeline Manager database and commits the data. If there is a failure, the utility rolls back the data and displays an error message.

Deleting Data from a Pipeline Manager Database

> **Note:**
>
> Make sure the utility is connected to the Pipeline Manager database. See "Connecting LoadIfwConfig to the Pipeline Manager Database".

To delete data from a Pipeline Manager database:

1. Create an XML file that specifies the data to delete. The file includes the table from which to delete the objects and, optionally, the criteria that the objects must meet. For information, see "About Specifying the Data to Extract".
2. Test the XML file by running the **LoadIfwConfig** utility with the **-r** or **fetch** parameter. Verify that the output file lists the correct objects to delete. See "Extracting a Subset of Database Objects with LoadIfwConfig" for more information.
3. Go to the *pipeline_home*/**tools/XmlLoader** directory.
4. Enter the following command:

   Interactive mode

   ```
   LoadIfwConfig
   read InputFile
   delete
   ```

   where:

   - *InputFile* specifies the name and location of the file that lists the objects to delete. This is the file that you created in step 1.

   Non-interactive mode

   ```
   LoadIfwConfig  -p[f] -i InputFile
   ```

   where:

   - **f** turns off the delete confirmation message.
   - *InputFile* specifies the name and location of the file that lists the objects to delete. This is the file that you created in step 1.

   > **Note:**
   >
   > For more information about the utility's syntax, see "LoadIfwConfig".

The utility deletes the specified database objects.

10

Creating iScripts and iRules

This chapter describes how to use the Application Programming Interface (API) to create custom Oracle Communications Billing and Revenue Management (BRM) iScript and iRules modules to process event detail records (EDRs) in the pipeline.

For information about the iScript functions, see "About iScript Functions".

About iScripts

iScript is a script language for analyzing and manipulating usage events. You can access any field within the usage structure and modify it. Using an iScript, you can implement and configure new guiding rules, mapping scenarios, discounting structures, interfaces to user-defined database objects and to reference data, and operating report files.

iScript is the Pipeline Manager script programming language that can be used to customize Pipeline Manager. It can be used for:

- Mapping operations that are not covered by standard modules.
- Adjusting values, such as normalizing phone numbers and IP addresses.
- Evaluating additional statistical data.
- Firing events.
- Performing splitting operations.
- Performing customer-specific pre- or post-rating or enrichment functionality.

You use iScripts to perform the same operation on every EDR. For example, if you have to enrich or normalize data in EDRs or if you have to evaluate EDRs before splitting them.

iScript makes it easier than C or C++ programming languages to implement custom functionality and business policies.

About iRules

An iRule consists of rule items that contain a number of conditions and a script to run when a condition is fulfilled. Only the script for the first valid rule item is run during the rule evaluation.

You create the scripts to be run by using pattern-matching evaluation (for example, wildcards and logical expressions) as well as normal comparison operators (for example, equal to, less than, and greater than).

You use iRules when a function should be run only on certain EDRs or under certain conditions. The rule-based engine offers an optimized method for defining logical conditions and extensive string comparisons. Therefore, iRules offer a faster way of implementing rule-based conditions than writing an iScript, which can include many "if-else" constructs.

You group rules and rule items into rule sets. A rule set defines the set of rules to be used by a particular iRule module. You specify the rule set in the FCT_IRule module startup registry.

> **Note:**
>
> You can refer to only one EDR extension block in an iRule. For example, you cannot use DETAIL.ASS_GSMW_EXT and DETAIL.ASS_SUSPENSE_EXT extension blocks in the same iRule. Use an iScript instead to compare or evaluate fields from multiple EDR extension blocks.

Creating Rule Sets via Description Files

In addition to the database interface and the file interface, you can also use a description file to define a rule set. This is useful when you store the rule data in separate database tables or ASCII files. A rule set defined by a description file can only have one rule.

Descriptions for Data from an ASCII File

In the following example description file:

- The rule name in is **ClassTypeMap**
- The source data is stored in an ASCII file called **/data/data.txt**.
- The variables, represented by **${*N*}**, are replaced by values in the source data file.

```
RULE: ClassTypeMap
SOURCE: File
FILE: /data/data.txt
INIT_SCRIPT:
String code = edrString( DETAIL.SERVICE_CODE ) + edrString( DETAIL.SERVICE_CLASS );
CONDITION:
code =~ "${1}";
edrString( DETAIL.CALL_CLASS ) =~ "${2}";
edrString( DETAIL.CALL_TYPE ) =~ "${3}";
RESULT:
edrString( DETAIL.CALL_TYPE ) = "${4}";
edrString( DETAIL.CALL_CLASS ) = "${5}";
```

The following example source data file contains the values that replace the variables (represented by **${*N*}**) in the definition file.

- The source data file must contain at least the number of columns, separated by semicolons, as there are variables in the definition file.
- The column position corresponds to the variable number. For example, ${2} in the description file is replaced with the value in column 2 of the source data file.

```
CODE1;CC1*;CT*;NewCC1;NewCT
CODE2;CC2*;CT*;NewCC2;NewCT
CODE3;CC3*;CT*;NewCC3;NewCT
CODE4;CC4*;CT*;NewCC4;NewCT
CODE5;CC5*;CT*;NewCC5;NewCT
CODE6;CC6*;CT*;NewCC6;NewCT
CODE7;CC7*;CT*;NewCC7;NewCT
CODE8;CC8*;CT*;NewCC8;NewCT
```

Descriptions for Data from a Database Table

In the following example description file:

- The rule name is **CZT_MapRule**.
- The source data is retrieved from the **IFW_CLASSTYPEZONE_MAP** database table.
- The data is ordered first by the ZONEMODEL column and then by the RANK column.
- Each variable, represented by **${***COLUMN_NAME***}**, is replaced by the value in that database table column.

```
SOURCE: Database
RULE: CZT_MapRule
TABLE: IFW_CLASSTYPEZONE_MAP
ORDERBY: ZONEMODEL
ORDERBY: RANK
INIT_SCRIPT:
String code = edrString( DETAIL.SERVICE_CODE ) + edrString( DETAIL.SERVICE_CLASS );
CONDITION:
code =~ "${CODE}"
edrString( DETAIL.CALL_TYPE ) =~ "${CALLTYPE}";
edrString( DETAIL.CALL_CLASS ) =~ "${CALLCLASS}";
edrString( DETAIL.SERVICE_CODE ) =~ "${SERVICECODE}";
edrString( DETAIL.AOC_ZONE ) =~ "${ZONE_WS}";
edrString( DETAIL.CHARGED_ZONE ) =~ "${ZONE_RT}";
RESULT:
if ( length( "${NEW_CALLTYPE}" ) > 0 )
{
edrString( DETAIL.CALL_TYPE ) = "${NEW_CALLTYPE}";
}
if ( length( "${NEW_ZONE_RT}" ) > 0 )
{
edrString( DETAIL.CHARGED_ZONE ) = "${NEW_ZONE_RT}";
}
if ( length( "${NEW_ZONE_WS}" ) > 0 )
{
edrString( DETAIL.AOC_ZONE ) = "${NEW_ZONE_WS}";
}
```

Importing and Exporting Validation Rules

You use the Database Storage and Extraction Tool for Validation Rules to extract validation rules from the Pipeline Manager database and to import them in the Pipeline Manager database. Pipeline Manager uses these validation rules for the roaming incollect and outcollect processes.

This tool uses DBI and DBD drivers which are not part of the Pipeline Manager installation. You download these drivers from www.cpan.org and compile and install them separately.

The Database Storage and Extraction Tool for Validation Rules consists of the following scripts:

- The **db2irules.pl** script extracts validation rules from the Pipeline Manager database.
- The **irules2db.pl** script adds validation rules to the Pipeline Manager database.

About the Rule Set XML File

The Database Storage and Extraction Tool for Validation Rules extracts the validation rules from the database to a Rule Set XML file. When creating the XML file, it maps the Structured Query Language (SQL) tables from the database to the appropriate XML tag names. When you import rules and rule items in the database, the rank for each of these is determined by the order in which they appear in the Rule Set XML file. For example, the first rule row or rule item

row found in the XML file is inserted into the database with a rank of 1, the second one is inserted with a rank of 2 and so forth.

Table 10-1 shows a possible mapping:

Table 10-1 Possible Mappings for Validation

Pipeline Manager Database Table	Pipeline Manager Database Table Column	XML Tag	Parent XML Tag
N/A	N/A	<RULESET_ROW>	NONE
IFW_RULESET	RULESET	<RULESET_RULESET>	<RULESET_ROW>
IFW_RULELSETLIST	RULESET	<RULESET_RULESET>	<RULESET_ROW>
IFW_RULESET	NAME	<RULESET_NAME>	<RULESET_ROW>
IFW_RULESET	DESCRIPTION	<RULESET_DESCRIPTION>	<RULESET_ROW>
N/A	N/A	<RULE_ROW>	<RULESET_ROW>
IFW_RULESETLIST	RULE	<RULE_RULE>	<RULE_ROW>
IFW_RULE	RULE	<RULE_RULE>	<RULE_ROW>
IFW_RULEITEM	RULE	<RULE_RULE>	<RULE_ROW>
IFW_RULE	NAME	<RULE_NAME>	<RULE_ROW>
IFW_RULESETLIST	NAME	<RULE_NAME>	<RULE_ROW>
IFW_RULE	INIT_SCRIPT	<RULE_INIT_SCRIPT>	<RULE_ROW>
N/A	N/A	<RULEITEM_ROW>	<RULE_ROW>
IFW_RULEITEM	NAME	<RULEITEM_NAME>	<RULEITEM_ROW>
IFW_RULEITEM	CONDITION	<RULEITEM_CONDITION>	<RULEITEM_ROW>
IFW_RULEITEM	RESULT	<RULEITEM_RESULT>	<RULEITEM_ROW>

About the db2irules.pl Script

You use the **db2irules.pl** script to extract rule sets from the Pipeline Manager database to the Rule Set XML file. You can extract any number of rule sets at the same time. If you extract multiple rule sets, each rule set is written to a separate Rule Set XML file.

When **db2irules.pl** extracts rows from the database, the text of each database row is checked for the following XML reserved characters:

- Left angle bracket (<)
- Right angle bracket (>)
- Ampersand (&)

The XML parser uses these characters to find the start tags, end tags, and any references. It replaces them with the following HTML and XML flags:

- <
- >
- &

These flags make the file XML compliant, so that any XML parser can parse this file successfully.

About the irules2db.pl Script

Use the **irules2db.pl** script to insert a rule set from Validation Rules XML file into the Pipeline Manager database. You can insert only one rule set at a time. To load multiple rule sets into the database, run this script separately for each rule.

When **irules2db.pl** imports rows to the database, each rule and rule set row of the XML file is checked for the following HTML and XML flags:

- <
- >
- &

The XML parser uses these characters to find the start tags, end tags, and any references. It replaces the m with the following XML-reserved characters:

- Left angle bracket (<)
- Right angle bracket (>)
- Ampersand (&)

If you specify an invalid file name for the rule set, the **irules2db.pl** script displays an error and terminates. If the file exists and can be opened, the script opens a transaction to the database and starts inserting the rule sets. If any of the rule sets specified already exists in the database, the **irules2db.pl** script reports an error, rolls back the transaction, and terminates. The error message contains information on which row in the XML file caused the error.

Updating a Rule Set

You can update rule sets that already exist in the database in two ways:

- Extract, update, and import a rule set
- Replace a rule set with an updated version

Extract, update, and import a rule set

To update a rule set that already exists in the database, export, update, and reimport the XML file:

1. Export the rule set to a Rule Set XML file by using the **db2irules.pl** script. For more information, see "About the db2irules.pl Script".

   Example:

   ```
   db2irules.pl -u dbi:Oracle:orcl /home/data/CIBER_val
   ```

2. Delete the rule set from the database. Specify the name of the rule set and set the **-d** parameter.

   Example:

   ```
   db2irules.pl -d dbi:Oracle:orcl /home/data/CIBER_val
   ```

   The script deletes all rules of a rule set recursively.

3. Update the rule set in the XML file. For more information, see "About the Rule Set XML File".

4. Import the rule set into the database.

   Example:

```
irules2db.pl dbi:Oracle:orcl /home/data/CIBER_val_2002_07_01_17-15-39.xml
```

Replace a rule set with an updated version

To replace an existing rule set with an updated version that you didn't extract from the database, you can import the rule set and create a backup of the old rule set by using the **-f** parameter.

Example:

```
irules2db.pl -f dbi:Oracle:orcl /home/data/CIBER_val /home/data/backup
```

The **-f** parameter causes the **db2irules** extraction script to be invoked and extract the original file from the database before loading the new file. A backup file of the original rule set is stored in the specified backup location.

Supported iScript Data Types

iScript supports the following data types in Table 10-2:

Table 10-2 Supported iScript Data Types

Data Type	Description
Bool	For Boolean values TRUE and FALSE.
String	For 8-bit character strings of an unlimited length.
Long	For signed integer values in the interval from - 9223372036854775808 (2^63) to 9223372036854775807 (2^63-1).
Date	For date/time values in the interval from the 1901/01/01 00:00:00 to the 2037/02/05 00:00:00.
Decimal	For floating pointer to numbers with a precision of 26 digits.
File	As a handle for files.

> **Note:**
>
> Based on the data types listed above, you can also use hash and arrays.

Supported iScript Constants

iScript supports constants described in this section.

Constants for Normalizing National Access Codes

Table 10-3 lists the iScript constants used to normalize national access codes.

Table 10-3 National Access Code Normalizing iScript Constants

Constant	Type	Value	Description
NORM_NAC	String	"0"	"National access code"
NORM_IAC	String	"00"	"International access code array"

Table 10-3 (Cont.) National Access Code Normalizing iScript Constants

Constant	Type	Value	Description
NORM_IAC_STRING	String	"00"	"International access code string"
NORM_CC	String	NA	Country code array
NORM_CC_STRING	String	"49"	"Country code string"
NORM_MCC	String	"262"	"Mobile country code"
NORM_IAC_SIGN	String	"+"	"International country code sign"
NORM_NDC	String	"172"	"Network destination code"

Date Constants

Table 10-4 lists the Date constants.

Table 10-4 Date Constants

Constant	Type	Value	Description
MAX_DATE	Date	05.02.2037 23:59:59	Maximum value for the date. The date is stored as the number of seconds since 00:00:00. The MAX_DATE value is the last date that can be represented with a four-byte unsigned long, which is February 5, 2037. The internal representation is adjusted to the time zone of the system.
MIN_DATE	Date	01.01.1901 00:00:00	Minimum value for the date. The date is stored as the number of seconds since 00:00:00. The MIN_DATE value is January 1, 1901. The internal representation is adjusted to the time zone of the system.

Database Connection Constants

Table 10-5 describes the database connection constants.

Table 10-5 Database Connection Constants

Constant	Type	Value	Description
INVALID_CONNECTION	Long	-1	Invalid database connection handle
INVALID_RESULT	Long	-1	Invalid result handle
NO_MORE_RESULTS	Long	0	No more results
NO_MORE_ROWS	Long	0	No more rows
NEXT_ROW	Long	1	Next row to be retrieved from the database table
NEXT_RESULT	Long	1	Next result from db query

Decimal Constants

Table 10-6 lists the decimal constants.

Table 10-6 Decimal Constants

Constant	Type	Value	Description
INVALID_DECIMAL	Decimal	Decimal::initInvalidDecimal();	Invalid Decimal value

Decimal Rounding Constants

Table 10-7 lists the decimal rounding constants.

Table 10-7 Decimal Rounding Constants

Constant	Type	Value	Description
ROUND_PLAIN	Long	0	If the digit to the right of the specified decimal place is equal to or greater than 5, add one to the digit at the specified decimal place, then truncate any digits to the right. Otherwise, truncate all digits to the right of the specified decimal place.
ROUND_UP	Long	1	If the digits the right of the specified decimal place are non-zero, add one to the digit at the specified decimal place and truncate any digits to the right.
ROUND_DOWN	Long	2	Truncate all digits to the right of the specified decimal place.
ROUND_BANKERS	Long	3	If incrementing the digit at the specified decimal place results in an even number, increment it and truncate the digits to its right. Otherwise, truncate the digits to the right of the specified decimal place.

EDR Container Content Constants

Table 10-8 lists the EDR container content constants.

Table 10-8 EDR Container Constants

Constant	Type	Value	Description
EDR_UNKNOWN_CONT	Long	1	Unknown EDR content type
EDR_HEADER	Long	2	EDR header record
EDR_DETAIL	Long	3	EDR detail record
EDR_TRAILER	Long	4	EDR trailer record
EDR_START	Long	5	Service container that tells pipeline starts
EDR_STOP	Long	6	Service container that tells pipeline shutting down
EDR_BEGIN	Long	7	Service container that tells EDR processing begins
EDR_END	Long	8	Ser vice container that tells EDR processing ends
EDR_BEGIN_TRANSACTION	Long	9	Service container that tells transaction begins

Table 10-8 (Cont.) EDR Container Constants

Constant	Type	Value	Description
EDR_END_TRANSACTION	Long	10	Service container that tells transaction ends

EDR Container Characters Deletion Constants

Table 10-9 lists the EDR container content deletion constants.

Table 10-9 EDR Container Content Deletion Constants

Constant	Type	Value	Description
STRIP_LEADING	Long	True	Roguewave constant used to delete the special leading characters in an EDR string, for example, white spaces at the beginning of the EDR container.
STRIP_TRAILING	Long	True	Roguewave constant used to delete the special trailing characters in an EDR string, for example, white spaces at the end of the EDR container.
STRIP_BOTH	Long	True	Roguewave constant used to delete both the special leading and trailing characters in an EDR string, for example, white spaces at the beginning of the EDR container.

EDR Input State Constants

Table 10-10 lists the EDR input state constants.

Table 10-10 EDR Input State Constants

Constant	Type	Value	Description
EDR_INPUT_MISSING	Long	0	Not supplied in input data
EDR_INPUT_EMPTY	Long	1	Supplied with no value
EDR_INPUT_OTHER	Long	2	Other "uninteresting" input state

EDR Internal State Constants

Table 10-11 lists the EDR internal state constants.

Table 10-11 EDR Internal State Constants

Constant	Type	Value	Description
STATE_CLEARED	Long	0	EDR value is cleared
STATE_CONNECTED	long	1	EDR value is connected with a field from an input record
STATE_INITIALZED	long	2	EDR value is initialized
STATE_SET	long	3	EDR value is set
STATE_RESTORED	long	4	EDR value is restored
STATE_RESTOREDASSET	long	5	EDR value is restored as set

Table 10-11 (Cont.) EDR Internal State Constants

Constant	Type	Value	Description
CONTAINER_HEADER	Long	2	Header record descriptor in container
CONTAINER_DETAIL	Long	3	Detail record descriptor in container
CONTAINER_TRAILER	Long	4	Trailer record descriptor in container
CONTAINER_UNKNOWN	Long	1	Unknown record type in container

POID Constants

Table 10-12 lists the Portal object ID (POID) constants.

Table 10-12 POID Constants

Constant	Type	Value	Description
NULL_PPOID	POID	BAS::Identifier(0, "", 0);	Null POID pointer
INVALID_PPOID	POID	BAS::Identifier	Invalid POID pointer

TAM Transaction Constants

Table 10-13 lists the TAM transaction constants.

Table 10-13 TAM Transaction Constants

Constant	Type	Value	Description
TAM_NORMAL	Long	0	Normal transaction
TAM_RECYCLE	Long	1	Recycle transaction
TAM_RECYCLE_TEST	Long	2	Recycle transaction for testing
TAM_UNKNOWN	Long	-1	Unknown transaction type

Supported Regular Expressions

iScript supports the regular expressions listed in Table 10-14:

Table 10-14 Regular Expressions Supported by iScript

Expression	Description
.	Matches any single character except the newline character **\n**.
****	Introduces metacharacters, and as part of the escape sequences. For example, **\n**, a newline character, and *****, is a literal asterisk. You can use it with three digits representing a number between 0 and 255 (ASCII code) to get the corresponding character. For example, **\065** is matched to the character **A**.
[ ]	A character class which matches any character within the brackets. If the first character is a circumflex (^), it changes the meaning to match any character except the ones within the brackets. A dash inside indicates a character range. For example [0-9] means the same thing as [0123456789].

Table 10-14 (Cont.) Regular Expressions Supported by iScript

Expression	Description
{ }	Indicates how many times to match the previous pattern when the pattern has one or two numbers, e.g. A{1,3} matches one to three occurrences of the letter A.
*****	Matches zero or more copies of the preceding expression.
+	Matches one or more copies of the preceding expression. For example. [0-9]+ matches 1, 111, or 123456 but not an empty string.
?	Matches zero or one copy of the preceding expression. For example. -?[0-9]+ matches a signed number including an optional leading minus.
|	Matches either the preceding expression or the following expression.
( )	Groups a series of expressions into a new expression. Useful when building up complex patterns with *, +, and |.
!	Negates the complete expression. It must be the first character in the expression.

iScript Variable Declarations

The syntax of the variable declaration is similar to the C/C++ syntax:

```
Type Name [=Value];
```

As an option, you can declare variables to be constants:

```
const Type Name = Value;
```

For example:

```
Long x;
String serviceCode = "Tel";
const Decimal pi = 3.1415927
```

iScript Arrays and Hashes

All data types except File can be used in arrays or hashes:

```
Long a[ ]; // A normal array declaration
String dist [ ] [ ]; // A 2-dimensional string array
String ndc { } [ ]; // An associative array
String cli { } [ ]; // An associative array or arrays
```

You do not have to specify the dimension of the arrays and hashes. The data structures are resized automatically and are initialized by default values. For numerical values, this is 0; for strings, it is an empty string and dates become invalid.

You can access arrays and associative arrays in the following way:

```
a [3] = 4711;
ndc { "040" } = Hamburg;
cli { "Max Mueller" } [0] = "0171123456789";
cli { "Max Mueller" } [0] = "0177471111245";
```

Note:

If you use arrays and hashes in functions, clear them at the start of the functions, as in this example:

```
function myFunction

{

    String str[];

    arrayClear(str);

     // ...

}
```

Arrays and hashes aren't initialized at the start of functions. They behave like static variables. If they are not cleared in a function, they retain values from the last time the function was run.

iScript Function Declarations

A function declaration has the following syntax:

```
function [returnType] identifier [( parameter [,parameter...])]
```

where *returnType* is optional. If you do not specify any return type, the default is VOID. A function can have an unlimited number of arguments. You can use the basic types as return and parameter types.

For example:

```
function Long square (Long x)
{
  return x*x;
}
```

Note:

Avoid nesting functions. Nested functions can create unexpected results, as in this example:

```
function myFunction

{

    Long i = 5;

    // ...

    myFunction();

        // Here the variable i is assigned the value set in the nested fu
nction.

    // ...

}
```

iScript Control Structures

The syntax of the control structures is similar to the C++ syntax, but because there are no implicit type casts, the following expression is *not* valid in iScript:

```
if ( i ) ...
```

You must use explicit type casts:

```
if ( i != 0 )...
```

There are AND and OR operators for Boolean expressions. Empty statements in FOR loops are not valid in iScript and there is no increment operator.

For example:

```
for ( ; i<100; i++ )
```

in C++ has to be replaced with

```
for ( i ; i<100; i=i+1)
```

in iScript.

iScript Function Blocks

A function block that is followed by a control structure must be enclosed in curly braces { }. This is also true if only one statement is in the function block.

For example:

```
if ( (edrString ( DETAIL.RECORD_TYPE) = = "H"))
  {
    logStdout ("Header detected\n");
  }
else
  {

  }
```

Using iScript Switch Statements

iScript provides switch statements for String values, Long values, and regular expressions. The syntax of the switch statement is similar to the C syntax. Follow these rules when including switch statements:

- Specify only one statement per **case** label.
- Use a statement block and enclose several statements between curly braces ({ }).
- Terminate every **case** label by a **break** statement. Otherwise the statement of the following **case** label is also run.
- For *regular expressions*, use the **regExprSwitchCase** statement instead of **switch**.

Examples for Switch Statements

Switch statements (Long)

```
switch ( edrLong ( DETAIL.RECORD_LENGTH ) )
{
  case 104:
    logStdout ( "Header record!" );
    break;
  case 685:
    {
           detail = detail + 1;
           logStdout ( "Detail record" );
    }
    break;
  default:
    logStdout ( "unknown record type" );
}
```

Switch statements (String)

```
switch ( edrString ( DETAIL.ERROR_TEXT) )
{
    case "ERR_DATETIME":
       logStdout ("invalid date/time" );
       break;
    case "ERR_NOIMSI":
       logStdout ("IMSI not specified");
    case "":
       logStdout ("no error detected");
       break;
...
}
```

Switch statements for regular expressions

```
regExprSwitch ( edrString ( DETAIL.A_NUMBER ) )
{
   case "0049171.*":
      logStdout ( "D1-call\n" );
      break;
   case "0049172.*":
      logStdout ( "D2-call\n" );
      break;
}
```

Including Other iScript Source Files in Your iScript

You can use the **include** statement in your iScript to include other iScript source files.

Use the following syntax and specify each **include** statement in a separate line:

```
include "iScriptFile.isc";
```

Before an iScript is compiled, a preprocessor evaluates and processes the **include** statements. If the included iScript file has an absolute path, the preprocessor tries to include the file with the absolute path. Otherwise, the preprocessor uses a semicolon-separated list of include directories specified in the **ISCRIPT_INCLUDE** environment variable.

If the **ISCRIPT_INCLUDE** environment variable is not set, the preprocessor uses only the current directory as the input directory. If the environment variable is set, it does not contain the current working directory by default. You must explicitly add the current working directory to the list by using a dot (**.**) as the path. For example:

bash and **sh:**

```
export ISCRIPT_INCLUDE="/home/integRate/iscript include;/usr/iscript;."
```

csh and **tsch:**

```
setenv ISCRIPT_INCLUDE "/home/integRate/iscript_include;/usr/iscript;."
```

About iScript Functions

You use the iScript functions to perform a variety of operations. See "Pipeline Manager iScript Functions".

About Special iScript Functions

Pipeline Manager includes a basic iScript interpreter and an extended interpreter that calls special iScript functions on external events, such as **onHeaderEdr** or **onDetaiEdr.** These special functions are function hooks that you can use to implement any actions you want to perform at specific situations, such as when a transaction is rolled back, during EDR processing.

For example, to perform custom actions when a transaction rolls back, you can include the following function block in an iScript:

```
Function onRollback
{
  logStdout("onRollback() \n");
  /* Define rollback-related actions here. */
}
```

Pipeline-Related Function Hooks

Use the functions listed in Table 10-15 to perform the actions you want at various stages of the pipeline process:

Table 10-15 Pipeline-Related Functions

Function	Description
BEGIN	Called when Pipeline Manager starts and after the iScript is compiled.
END	Called when Pipeline Manager is shut down.
onMessageReceived	Deprecated.
onStartEdr	Called when a pipeline starts. It is called once in the life of a pipeline process.
onStopEdr	Called when a pipeline stops. It is called once in the life of a pipeline process.

EDR Processing-Related Function Hooks

Use the functions listed in Table 10-16 to perform the actions you want during various stages of EDR processing:

Table 10-16 EDR-Processing Related Functions

Function	Description
onBeginEdr	Called when a BEGIN EDR container for each file opened in a transaction passes through a module.
onBeginTransaction	Called when a BEGIN_TRANSACTION EDR container passes through a module.
onDetailEdr	Called when a DETAIL EDR container passes through a module.
onEndEdr	Called when an END EDR container for each file opened in a transaction passes through a module.
onEndTransaction	Called when an END_TRANSACTION EDR container passes through a module.
onHeaderEdr	Called when a HEADER EDR container passes through a module.
onInvalidDetailEdr	Called when an EDR with invalid detail record is received by a module.
onTrailerEdr	Called when a trailer container passes through a module.

Input Grammar-Related Function Hooks

Use the function hooks listed in Table 10-17 in the input grammar to run the actions you want when the input module parses a stream:

Table 10-17 Input Grammar-Related Functions

Function	Description
onParseEnd	Called when the input module finishes parsing a stream.

Table 10-17 (Cont.) Input Grammar-Related Functions

Function	Description
onParseError	Called when the input module encounters an error while parsing a stream.
onParseStart	Called when the input module starts processing a stream.
streamIsEmpty	Called when an empty stream is encountered.

Transaction-Manager Related Function Hooks

Use these functions listed in Table 10-18 to perform the actions you want during a transaction.

Table 10-18 Transaction Manager Related Functions

Function	Description
onCancel	Called when a transaction needs to be cancelled.
onCommit	Called when the transaction manager notifies that a transaction is committed.
onPrepareCommit	Called when the transaction manager sends a request to prepare to commit a transaction.
onRollback	Called when a transaction needs to be rolled back.

About iScript Flist Extension Functions

You use the iScript flist extension functions to manipulate data in flists. For example, you can use the functions to retrieve information from an flist so that you can add it to the EDR container. You can also use the functions to add data taken from an EDR to an flist.

There are functions for creating flists, retrieving data from fields, setting field values, setting and unsetting the current array, deleting fields and arrays, and retrieving error text.

Here is a simple example of moving data from an EDR to an flist. Suppose you have the following EDR data block:

```
DETAIL.ASS_DATA
   String NAME
   Decimal VALUE
   Long QUANTITY
```

You can convert that EDR block to an flist in this format:

```
0 PIN_FLD_ARRAY   ARRAY
1   PIN_FLD_STRING   STRING
1   PIN_FLD_DECIMAL   DECIMAL
1   PIN_FLD_INT   INT
```

The following is an example of iScript code to convert the EDR block to an flist:

```
fListPushElem( "PIN_FLD_ARRAY", 0 );
fListSetString( "PIN_FLD_STRING", edrString( DETAIL.ASS_DATA.NAME, 1 ) );
fListSetDecimal( "PIN_FLD_DECIMAL",edrDecimal( DETAIL.ASS_DATA.VALUE, 1 ) );
fListSetLong( "PIN_FLD_INT", edrLong( DETAIL.ASS_DATA.QUANTITY, 1 ) );
```

```
fListPopElem();
```

Improving Pipeline Performance in Custom iScripts and iRules

This section provides some guidelines to reduce pipeline startup time and memory usage in your custom iScripts and iRules.

When the pipeline framework loads an iRule, a finite state machine (FSM) is built in memory. The number of objects the FSM creates in memory affects the pipeline startup and memory usage. FSMs create decision trees in memory at startup-time which affect pipeline startup, but decision trees help the pipeline work efficiently at run-time.

To reduce pipeline startup time and memory usage, follow these guidelines while creating iRules:

- Split the database entries because the number of rows in the loaded table affects memory usage and startup time of Rules.
- When writing rules, review the condition and reduce the number of compare patterns.
- Keep the regular expressions as simple as possible.
- If you have complex rules, place them at the beginning of an iRule.
- To improve processing performance, split big iRules if you have CPUs to allocate for new threads.

To improve pipeline processing performance, follow these guidelines when creating iScripts:

- If you can use an iRule instead of an iScript, use an iRule.
- Keep iScripts simple.
- Whenever possible, use standard modules instead of creating custom modules.
- Avoid database read access from the iScript when processing EDRs. Instead, load data during startup into hashes, and use that data when processing EDRs.
- Avoid functions that duplicate EDRs. They are performance intensive.
- Avoid writing to a database within an iScript. If you do, make sure you handle transactions properly.

Part II

Setting Up Pricing for Real-Time Rating and Pipeline Manager

This part describes how to configure real-time and pipeline rating in an Oracle Communications Billing and Revenue Management (BRM) system when Pricing Center and Pipeline Manager is used for pricing and rating. It contains the following chapters:

11

About Creating a Price List

This chapter presents an overview of how you can create an Oracle Communications Billing and Revenue Management (BRM) price list in Pricing Center to specify how to charge for your services.

Topics in this document:

Before reading this chapter, you should have a basic understanding of BRM charging. See "Overview of Charging" in *BRM Concepts*.

About Price Lists

You create a price list to define how much to charge for your services. A *price list* consists of several components:

Plans

You use *plans* to offer your services to customers. For example, if your company provides Internet access and email, your plans might include:

- A plan that offers wireless telephony.
- A plan that offers only Internet access.

- A plan that offers Internet access and email.

During account creation, your customers are presented with a list of plans (for example, “Unlimited Internet Access" or “500 free wireless minutes with unlimited roaming").

Deals

A plan consists of one or more deals. You use deals to define different ways to charge for services. For example, you can offer two different deals for Internet access:

- A deal that includes a setup fee.
- A deal that has no setup fee.

Each deal is typically associated with a specific service. For example:

- A plan that offers only Internet access includes only Internet access deals.
- A plan that offers Internet access and email includes two deals, one for Internet access and one for email.

In addition, you can purchase more than one instance of the same deal and have it applied to a single account or service.

Products

A deal consists of one or more products. You use products to package and organize the rates that define how much to charge for your services.

Your products might include:

- A setup fee.
- A monthly subscription fee.
- Usage fees for telephone calls.

  Usage fees can be rated in real time or in a batch rating pipeline.
- An offer profile

  An offer profile is associated with a product or discount.

Offer Profiles

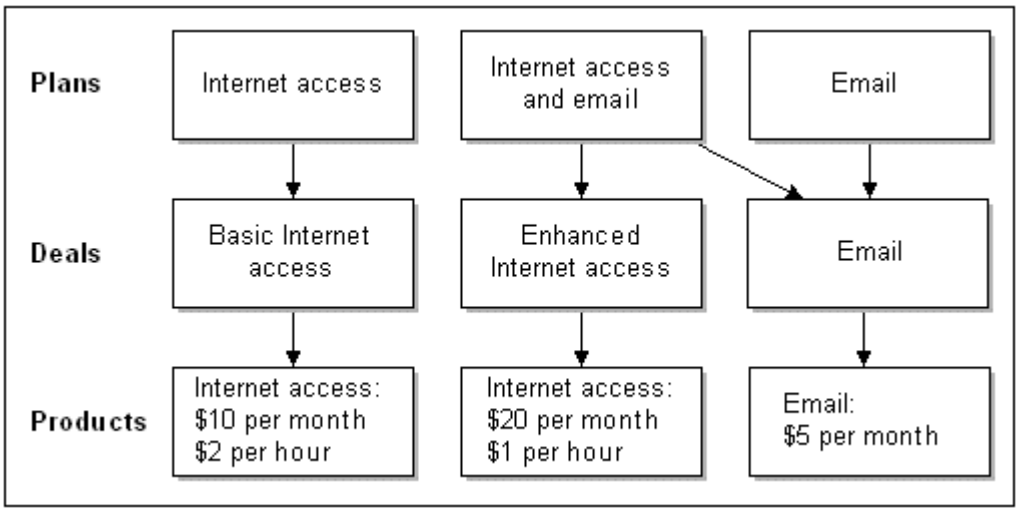
You use *offer profiles* in association with products and discounts to provide policy-driven provisioning of services. Each offer profile consists of the following data:

- A unique name
- An array of policy labels, with each policy label composed of rate tiers

Figure 11-1 shows how a simple price list is organized. In this example:

- The Internet access plan includes the Basic Internet access deal.
- The Email plan includes the Email deal.
- The Internet access and email plan includes the Enhanced Internet access deal and the Email deal.

Figure 11-1 Simple Price List



In this figure, notice that you specify how much to charge for your services at the lowest level, in the rates that are included in products. Therefore, when you specify how much to charge for your services, you start by creating products and rates.

What Is Rating?

Rating is the process of measuring customer activity, determining how much to charge for it, and adding the charge to the customer's account balance.

For example, if Internet access is rated at $1 per hour:

1. A customer has a 2-hour dialup session.
2. BRM rates the session at $1 per hour.
3. The customer's account balance increases by $2.

In a typical business, thousands of customers log in daily, generating millions of interactions that must be managed and charged for. BRM manages those interactions by creating and storing billable events.

About Billable Events

An *event* is a record in the BRM database of a customer or administrative action. For example:

- When a customer logs in to a dialup session, BRM creates a session event, which stores data about the session, such as the start time and end time.
- When a customer service representative (CSR) changes a customer's password, BRM creates an activity event, which stores information such as the identification of the CSR who made the password change.

When you set up your price list, you define which events you want to charge for. These events are called *billable events*. To determine how much to charge a customer for a billable event, BRM rates the event.

How BRM Rates a Billable Event

You can rate billable events in two ways:

- *Real-time rating* monitors and rates service usage as it happens, such as Internet access. Real-time rating is performed by rating opcodes.
- *Batch rating* rates events that have been recorded in files, such as telephony call detail records (CDRs). Batch rating is performed by Pipeline Manager or by Universal Event (UE) Loader. Most telco services use Pipeline Manager for batch rating.

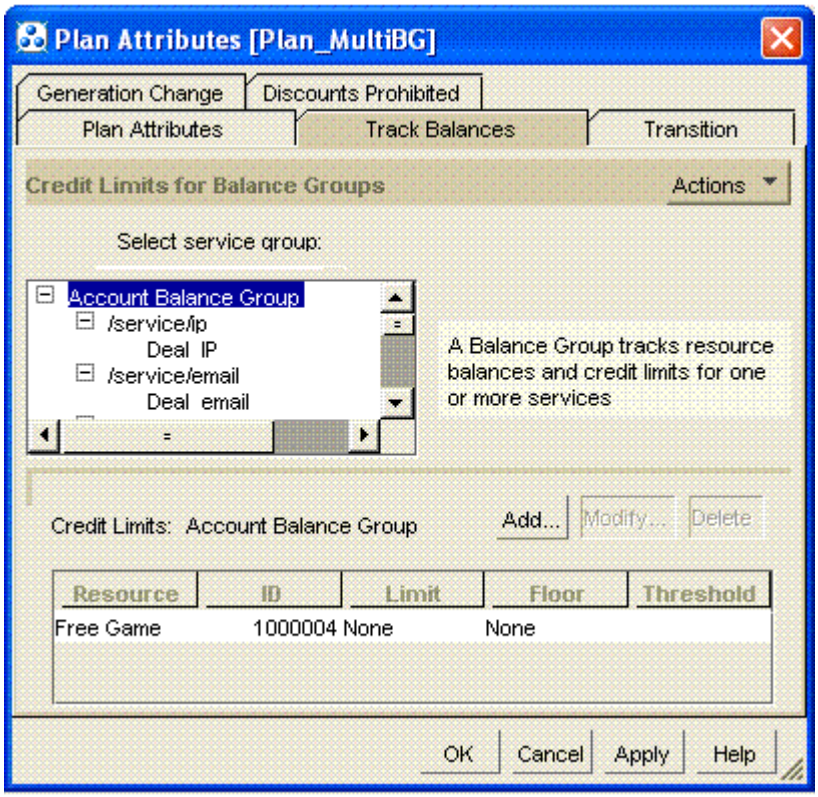
About Real-Time Rating

Real-time rating is used for services such as dialup access and email. For example, to rate Internet access in real time:

1. BRM determines the length of the session by comparing the start time and end time.
2. BRM applies a charge to the amount of time. For example, if you specify to charge $1 per hour for Internet access, a 10-hour session costs $10. This charge is called the *balance impact*.
3. BRM adds the total charge for the event to the customer's account balance.

Figure 11-2 shows how a billable event is captured, rated, and recorded in the BRM database:

Figure 11-2 Track Balances Tab



About Pipeline Batch Rating

Pipeline batch rating is typically used for rating telephony services, where large amounts of events are recorded in files. For example, to rate telephone calls:

1. Pipeline Manager reads the data from CDRs to determine the time and duration of the calls.
2. BRM applies a charge to the amount of time. For example, if you specify to charge 10 cents per minute, a 10-minute call costs $1. This charge is called the *balance impact*.
3. BRM adds the total charge for the event to the customer's account balance.

> **Note:**
>
> You can also use UE Loader to load data from event log files as billable events and rate them using real-time rating. The event log files can include data from Web servers or other services that record events in files. When the events are loaded, BRM rates the events and updates the customers' account balances. For more information, see *BRM Loading Events*.

For more information, see "About Pipeline Rating".

About Different Types of Rates

Before you set up your price list, you must determine which types of rates to use:

- *Usage rates* rate service usage, such as telephone calls or Internet dialup sessions.
- *Cycle rates* rate recurring fees that are not generated by usage (for example, a monthly subscription fee). See "About Cycle Rates".
- *Purchase rates* charge for nonrecurring fees, such as setup fees. Purchase events are created when a customer purchases a product.
- For account and product cancellations, use **cancel events**. Cancel events occur only when a product is canceled. Cancel events are not affected by how much a customer uses a service.

  When you close an account, all the products in the account are canceled. Therefore, cancel events are generated for all the canceled products. You do not need to cancel the products first to generate cancel events.

> **Note:**
>
> You use pipeline rating only for rating usage events. You can use real-time rating to rate all types of events.

About External and Internal Events

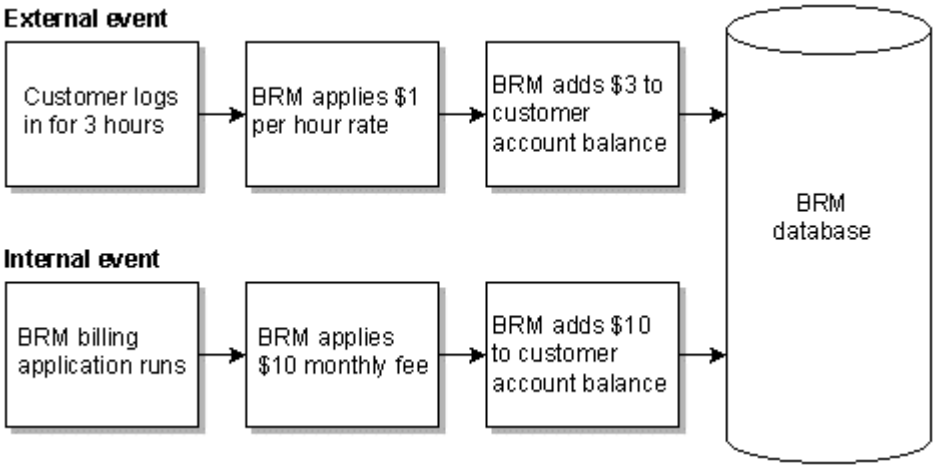
An *external* billable event is a usage event created by the customer outside of the BRM system (for example, when a customer makes a telephone call).

An *internal* billable event is an event that is not generated by the customer, but is instead generated by the BRM system. For example, on the customer's billing day, BRM creates a monthly fee event. This event is rated by a cycle rate.

Different rates are used to rate external and internal events.

Figure 11-3 shows how BRM rates external and internal events. The external event was triggered by a dialup session. The internal event was triggered by running a billing utility.

Figure 11-3 External and Internal Event Rating Flow



About Cycle Rates

To charge subscription fees, such as monthly payments for having an account, you create rates for cycle events.

There are three types of cycle events: cycle forward, cycle arrears, and cycle forward arrears.

About Cycle Forward Events

Cycle forward events charge a fee for future service. For example, when a customer pays the monthly cycle forward fee for a cell phone service, the customer pays for the coming month. With a yearly cycle forward fee, the customer pays for the entire year in advance.

Cycle forward events typically occur at the end of a billing cycle to charge the customer for the upcoming billing cycle, but you can define your own flexible cycles.

There are five cycle forward event types to support: monthly, bimonthly, quarterly, semiannual, and annual fees. Multi-month cycle forward events can occur on any date; they do not have to be synchronized to the start or end of a month.

> **Note:**
>
> You can also configure support for cycles of any length (for example, weekly or every five months).

About Cycle Arrears Events

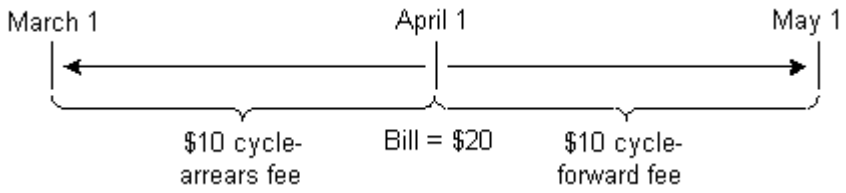
Cycle arrears events occur at the end of the month to charge the customer for the past month. When a customer pays a cycle arrears fee, the customer pays for the month that has already occurred.

> **Note:**
>
> BRM does not support multi-month cycle arrears fees.

In Figure 11-4, the customer owns two plans, each with a cycle fee. The bill created on April 1 includes:

- A $10 cycle arrears fee for March
- A $10 cycle forward fee for April

Figure 11-4 Cycle Arrears and Cycle Forward Fees



About Cycle Forward Arrears Events

Cycle forward arrears events occur at the beginning of the month to charge customers for the upcoming month, but the cycle fees are not billed until the end of the month when billing is run. When a customer pays a cycle forward arrears fee, the customer pays for the month that has just passed.

> **Note:**
>
> BRM does not support multi-month cycle arrears fees.

BRM adds cycle forward arrears fees to account balances as unbilled revenue at the beginning of the cycle. This enables you to recognize unbilled cycle arrears fees when you run G/L reports before the cycle ends. For more information, see the discussion about collecting general ledger data in *BRM Collecting General Ledger Data*.

Cycle forward arrears fees are stored in cycle forward arrears items. The cycle forward arrears event is assigned to the item that belongs to the next accounting cycle. This way, the fee is tracked in the account balance for the current cycle, but it is not billed until the end of the cycle.

About Using Delayed Billing

Because cycle forward arrears fees are assigned to the item that belongs to the next cycle, you must use delayed billing even if you do not need a delayed billing period.

When you use delayed billing, BRM tracks events in the current cycle that will be billed in the next accounting cycle. When the bill item for the next accounting cycle is created, BRM then assigns the tracked events to that bill item.

If you do not need a delayed billing period, you can set the value of the delayed period to **0**.

For information about configuring delayed billing, see the discussion about setting up delayed billing in *BRM Configuring and Running Billing*.

How BRM Creates Cycle Events

Cycle events are internal events created when you run the **pin_bill_accts** billing utility. For example, if the **pin_bill_accts** utility finds that an account's billing date is due, and the customer has signed up for a plan that includes a monthly cycle forward fee, BRM creates a Monthly Cycle Forward Event.

For information about accounting cycles, see the discussion about accounting and billing cycles in *BRM Configuring and Running Billing*.

Benefits of Using the Cycle Event Types

The advantage of using a cycle arrears event is that all of the usage and cycle fees for a particular month are included in the same bill.

The advantage of using a cycle forward event is that you collect payments sooner. Most implementations use cycle forward events.

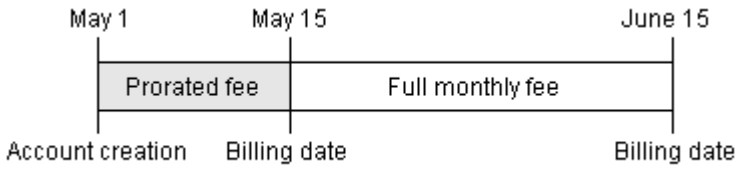
The advantage of using a cycle forward arrears event is that all of the usage and cycle fees for a particular month are included in the same bill, *and* the cycle fee is recorded in the G/L when a G/L report is run before the fee is billed.

Allowing Cycle Fees to Be Prorated

Cycle events are not related to how much a customer uses a service. However, you can enable cycle charges to be prorated. By default, BRM prorates cycle fees in the following cases:

- When a customer registers for an account and the account billing day is not the same day that the account was created. In this case, the customer is billed for the partial month that falls between registration and the first billing date as shown in Figure 11-5.

Figure 11-5 Prorating Cycle Fees



In this example, a prorated cycle forward fee is charged on May 1, when the account is created to pay for account ownership from May 1 through May 15.

- When a customer changes the billing day of month (DOM) and a partial month is created as a result.

- When a customer cancels a product before the end of the billing cycle.

You can specify if a cycle fee should be prorated, if the complete fee should be applied, or if no fee should be applied.

When you specify that a fee can be prorated, you specify which resources can be prorated. For example, you can prorate currency, but not free Internet access hours.

For details on proration, see the discussion about calculating prorated cycle fees in *BRM Configuring and Running Billing*.

Applying Multiple Rates to the Same Event

You can apply multiple rates to the same event by creating multiple rates within a rate plan. For example:

- You can apply different time of day rates to a single type of dial-up event. See:
  - Real-Time Rating Based on Date and Time
  - Rating by Date and Time with Pipeline Manager
- You can apply a different rate based on the quantity that has already been rated. See "Real-Time Rating Based on Event or Resource Quantity" for more information.
- If you are defining rates and creating your price list using the XML Pricing Interface, you can apply a different rate based on event date, purchase date, or service instantiation date. This lets you use different rates to create different versions of the same product. See "Using Date Ranges for Versioning" for more information.

Ways to Rate Events

When you create your price list, you can use many different ways to define rates (for example, which data you measure, how to measure the data (duration or occurrence), and time of day).

About Ratable Usage Metrics

You can rate an event based on any data captured in the event. For example, you can measure and rate how long a session is or measure and rate the number of bytes downloaded. The event data that you use to rate an event is called the *ratable usage metric*, or RUM. Common RUMs are:

- Duration. You rate based on how long a usage event was.
- Occurrence. You rate based on how many events occurred, independent of their duration.

You set up RUMs for pipeline batch rating and real-time rating. For information, see the following:

- Pipeline batch rating: "Setting Up Real-Time and Pipeline Pricing and Rating".
- Real-time rating: "About Setting Up Rums for Real-Time Rating".

About Applying Multiple RUMs to Rate an Event

By default, BRM rates an event by using a single RUM that is specified in the product's usage map. You can also set up your products to rate an event based on multiple RUMs.

For example, you can set up a product to rate fax events based on the following RUMs:

- The number of bytes transmitted

- The number of pages transmitted
- The duration of the fax session

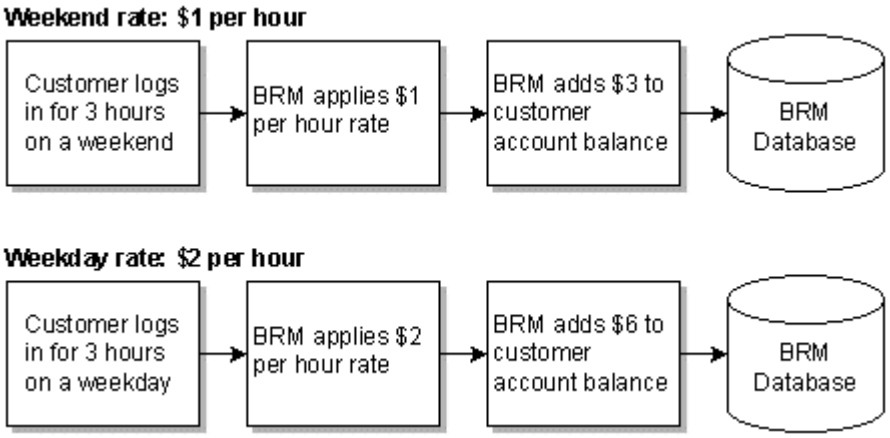
For more information, see "Real-Time Rating Based on Multiple RUMs".

About Rating Based on Date and Time

You can use different rates depending on the date and time that the event occurred. For example, you can apply one rate on weekdays and another rate on weekends.

Figure 11-6 shows how the same event can be rated differently based on when it occurred:

Figure 11-6 Rating Based on Date and Time



You can use pipeline batch rating and real-time rating to rate events based on date and time. For information, see the following:

- Pipeline batch rating: "Rating by Date and Time with Pipeline Manager".
- Real-time rating: "Real-Time Rating Based on Date and Time".

About Resources

When you create rates, you can specify different resources to use for the balance impact. A resource is an asset of economic value, such as US dollars and free Internet access hours.

- You must create currency resources for the system currency and the account currency that you support. For example, if you have Canadian customers, you must create the Canadian dollar resource.
- You must create noncurrency resources if your rates include noncurrency balance impacts such as free minutes or Internet access hours. For example, when you provide free Internet hours, you need a resource to create balance impacts for the access hours.

You set up resources for pipeline batch rating and real-time rating. For information, see the following:

- Pipeline batch rating: "Rating by Date and Time with Pipeline Manager".
- Real-time rating: "Real-Time Rating Based on Date and Time".

About Rounding Resources

You round resources for various reasons (for example, to increase the accuracy of rating and discounting, display rounded amounts on bills, and show the correct decimal value for different types of currencies such as dollars and yen).

BRM enables you to round resources based on the type of resource or currency, the type of event such as purchase, usage, and discount events, and the process that performs the rounding such as rating, discounting, and billing. These options enable you to set up rounding in specific ways. For example, you can round up to a precision of six decimal places for rating usage events, round down for discounting those usage events, and round to the nearest two decimal places when billing the usage.

You set up rounding rules for pipeline batch rating and real-time rating. For information, see the PDC Online Help or Pricing Center Help.

About Setting Rules for Resource Consumption

You can specify the order in which resource sub-balances are consumed. For example, if a customer has several groups of free minutes that expire at different times, you use consumption rules to indicate which minutes to use first, based on the validity period start time and end time.

You set up consumption rules for pipeline batch rating and real-time rating. For information, see "Specifying the Order in Which Resource Sub-Balances Are Consumed".

About Rating Based on Event Attributes

With attribute-based rating, you can specify different event attributes to consider for rating. For example:

- Telephony events include two event attributes that record the call origin and destination. If you offer a telephony service, you can rate call events according to call origin and destination.
- Telephony events also include an event attribute that records the type of call (for example, a call made with a calling card). You can use this attribute to rate calls made with a calling card differently from other calls.

You can use pipeline batch rating and real-time rating to rate events based on event attributes. For information, see the following:

- Pipeline batch rating: "Rating by Date and Time with Pipeline Manager".
- Real-time rating: "Real-Time Rating Based on Date and Time".

About Rating Based on Event or Resource Quantity

You can rate events based on the quantity that has already been rated. For example:

- 10 cents per minute for the first 30 minutes of usage.
- 6 cents per minute for the next 60 minutes of usage.
- 4 cents per minute for usage over 90 minutes.

You can use pipeline batch rating and real-time rating to rate events based on event or resource quantity. For information, see the following:

- Pipeline batch rating: "Rating by Date and Time with Pipeline Manager".
- Real-time rating: "Real-Time Rating Based on Date and Time".

About Products

When you create your price list, you organize rates into products. A *product* is the basic unit of your price list. Each product defines a set of events, usually associated with a service, and the balance impacts that are applied when those events occur.

Specifying Rates in Products

When you create a product, you specify basic information about the product, such as the name and description. You also specify the following rating information:

- The events to rate (for example, monthly cycle forward or IP Dialup events). You can rate more than one type of event in a product.

> **Note:**
>
> A single product cannot include multiple cycle events that have the same frequency and type of balance impact, such as a cycle fee. For example, if you add a monthly cycle forward event to a product, you cannot also add a monthly cycle arrears or monthly cycle forward arrears event to the same product. Instead, Oracle recommends that you create separate products for each cycle forward event.

- The service that the product applies to. For example, to create a product for rating an IP Dialup service, you use **/service/ip**.
- For each event, the ratable usage metric (RUM). For example:
  - For cycle forward events, rate by occurrence.
  - For dialup events, rate by duration.
- For each event, the resource used for the balance impact. For example:
  - To charge money, the resource might be US Dollars.
  - To give free hours, the resource might be Dialup hours.
- The balance impact of each event (for example, $1 per hour).

For example, a product for rating an IP Dialup service might include the rating information shown in Table 11-1:

Table 11-1 Rating Information for Dialup Service

Event: Monthly Cycle Forward Event	Event: IP Dialup Event
• RUM: Occurrence • Resource: US Dollars • Balance impact: $20 per occurrence	• RUM: Duration • Resource: US Dollars • Balance impact: $1 per hour

In addition to the rating information shown in this example, you can supply additional information, such as how to calculate taxes, the dates and times that a rate is valid, how to rate based on quantity, and if a rate is discountable.

You can create products in many different ways to accommodate many pricing scenarios.

Using Products to Rate Different Services

When you offer multiple services, you typically create products for each service. Table 11-2 shows an example:

Table 11-2 Products and Services

IP Access Product	Email Product	Fax Product
$10 purchase fee	No purchase fee	$10 purchase fee
$10 monthly charge	$5 monthly charge	$5 monthly charge
$1 per hour usage fee	$5 per month for an extra mailbox	No usage fee

You can create products for different services in a single plan, but a product cannot be used to rate more than one service.

Using Products to Charge Different Amounts for the Same Service

You can use products to charge for the same service in different ways. You can do this by rating combinations of internal and external events. Table 11-3 shows an example.

Table 11-3 Rating Combinations

Standard Product	Low Usage Product	Unlimited Usage Product
$20 purchase fee	$10 purchase fee	$20 purchase fee
$10 monthly charge	$5 monthly charge	$20 monthly charge
$1 per hour usage fee	$2 per hour usage fee	No usage fee

Associating Products with Offer Profiles

To provide policy-driven provisioning of services, you associate products with offer profiles. For example, if you create an offer profile called *Data Pro Unlimited*, you use that name as the provisioning tag for your product. The resource tracking is made possible when you configure the resource id for the resource named in the offer profile (for example, **100009** for *Megabytes Used*) as the resource counter in the product.

Associating Products with Accounts

You can create products that associate rates with accounts. For example, you can create a product that includes monthly charges for an account independent of the services that a customer owns as shown in Table 11-4:

Table 11-4 Associating Products and Accounts

Account Product	IP Access Product	Email Product
No purchase fee	$10 purchase fee	$10 purchase fee
$10 monthly charge	No monthly charge	No monthly charge
$5 cancel fee	$1 per hour usage fee	No usage fee

Using Products to Handle Complex Rating

You can rate a single event in different ways in the same product. Table 11-5 shows an example.

Table 11-5 Complex Rating with Products

Simple Product	Complex Product
$10 purchase fee	Purchase fees: • $10 purchase fee if purchased after December • $5 purchase fee if purchased after October • No purchase fee if purchased before or during October
$1 per hour usage fee	Usage fees: • $1 per hour on weekends • $2 per hour on weekdays • First 20 weekend hours free • First 10 weekday hours free

Specifying Minimum Event Quantities for Rate Plans

You can specify a minimum event quantity to rate. For example, suppose the minimum quantity for the IP access rate is 120 seconds (2 minutes). IP access events of less than 120 seconds are rated as if they were 120 seconds long.

Specifying How to Round Event Quantities

You can specify how to round fractional event quantities. For example, if you round Internet access to the nearest second, and a customer connects to the Internet for 2 minutes and 39.76 seconds, the event is rated at 2 minutes and 40 seconds. Event quantities are always rounded up.

You can take advantage of event quantity rounding to charge for consistent blocks of time (for example, 5 cents for each 10 seconds of an IP telephony call).

> **Note:**
>
> Event quantity rounding is different from resource rounding. With resource rounding, you round the amount to charge. With event quantity rounding, you round the amount of usage. See "Setting Up Resources for Real-Time Rating".

About Organizing Rates in a Product

You organize rates in a product in a hierarchical structure. This enables you to create multiple components at each level; for example, you can rate multiple events in a single product and create multiple rates for each event.

About the Event Map

The event map specifies the events that are being rated in the product. For example:

- Monthly Cycle Forward Event
- IP Dialup Event

See "Mapping Events and Services".

About Rate Plans

You use rate plans to define how much to charge for an event. There are two separate but related kinds of rate plans. In the simplest terms, real-time rate plans are used for real-time rating and pipeline rate plans are used for batch rating.

> **Note:**
>
> If you use Pipeline Manager to rate events, you must use the same rate plan names in your price list and in the pipeline pricing configuration.

About Product Ownership

During account creation, the customer chooses a plan. A plan is a package of deals, each of which in turn is a package of products. Therefore, what the customer actually owns is not a plan, but a set of products.

If you create different products for the same service, different customers might use the same service, but pay different charges, based on the products that they own.

For example, you might have two customer accounts that use the same service, but have different products as shown in Table 11-6:

Table 11-6 Products and Accounts

Account 1	Account 2
IP product 1: • Service: IP Dialup • Cycle forward event: $20 • Usage events: $2	IP product 2: • Service: IP Dialup • Cycle forward event: $10 • Usage events: $1

When BRM rates events for these customers, the charges for Account 1 are different from the charges for Account 2, even though the services and the events being rated are the same type.

> **Note:**
>
> When you manage a customer's account, you manage their products. For example, you can change the product status or customize the product's pricing. When you create your price list, it is important to consider how the products will be managed after they have been purchased.

Configuring Full Day Proration

When you create a charge offer, set the rounding values for the following:

- **The validity period**: You can specify that the validity period starts at the purchase time or midnight of the day the product is purchased. Alternatively, you can set to use the systemwide setting in the CM **pin.conf** file.
- **The charging scale**: You can specify whether to charge for a full day or a partial day for the first day of the billing cycle. Alternatively, you can set to use the validity period setting.

  See "Setting Full Day Proration".

Specifying Which Services Are Rated by a Product

To define which services are rated by a product, you specify whether a product is associated with an *account* or with a *single service*.

- **Account**. This product applies only to an account. For example, you could charge a monthly fee just for having an account, regardless of the services. Even if all services are inactivated, the monthly charge would still be applied. You can also use an account-level product for an account sign-up fee.

  You might create account-level products if you offer several services that customers do not own for any length of time (for example, access to Web simulcasts). In that case, a customer might own the following products:

  – Account-level product: $20 monthly fee

  – IP Dialup service product: $10 per occurrence

- **Single service**. Usually, you associate a product with a service (for example, IP Dialup or Email). When you do this, the rates in the product are applied to that service only. It is easier to manage a price list when each product has a purchase level associated with a single service.

  When a product applies only to a single service, the product is inactivated when the service is inactivated, and charges for the service stop.

  You can associate as many products with the same service as you want.

> **Note:**
>
> Products must be associated with the same service as the deals that contain them. For example, if your product allows **/service/ip**, any deals containing that product must also be valid for **/service/ip**.

About Specifying the Events to Rate in a Product

You might need to charge different types of fees for a service, such as subscription fees and usage fees. When you create a product for a service, you define the types of fees to charge by specifying which events to rate. The group of events rated by a single product is called an *event map*.

For example, to charge purchase, subscription, and usage fees for a service, a product's event map includes:

- Product Purchase Fee Event
- Monthly Cycle Forward Event
- IP Dialup Event

You do not have to include all types of events in a single product. A product only requires only one event in the event map, so you could create individual products for each type of event that you want to rate.

How you choose which events are rated in a product depends in large part on the services you offer, but it also depends on how you manage your business and your customers. For example, you might change your pricing often, provide sponsored services, or provide numerous types of discounts. Before you finalize your price plan, run some discounting and customer management scenarios to determine if your products are organized in a way that supports your business.

Specifying Product Types

There are three types of products: item, subscription, and system.

- *Item products* contain rates that are applied only once (for example, a purchase fee). The only event type you can use for an item product is the purchase event.
- *Subscription products* contain rates that are applied on an ongoing basis (for example, usage charges and monthly charges).
- *System products* contain rates that can be applied to all products in your price list. For example, you could create products to charge for IP usage with various limitations for various customers. You could then create a system product to charge a default usage rate of $0.10 per minute for all your customers when those other products are not valid.

You specify product types when you use Pricing Center to create a product as shown in Figure 11-7:

Figure 11-7 Product Type in Pricing Center

Specifying General Product Information

To create a product, you specify a product name and description. In addition, you specify information about provisioning, taxes, priority, and time- and quantity-based validity ranges.

If you are using the XML pricing interface to create your price list, you can also define what happens if a customer purchases the same product more than once. By default it is purchased as a new product, but you can specify that it instead:

- Replaces the existing product
- Extends the existing product if it is purchased within a specified grace period after the existing product expires

See "Purchasing the Same Product or Discount Multiple Times" for more information.

Defining How Many Products a Customer Can Purchase

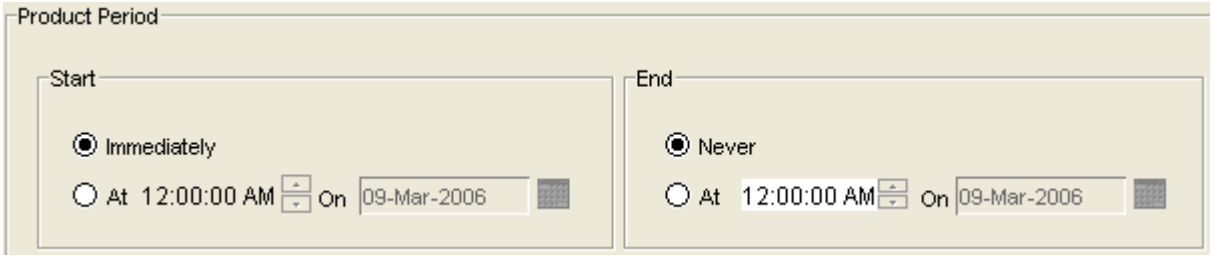
You can define the following quantity attributes:

- You can specify whether a customer can purchase part of a product for a reduced price. For example, if an IP fax product provides 100 pages faxed for $50, you can allow customers to purchase 50 pages for $25.
- You can define the minimum and maximum numbers of products that can be purchased. For example, if a product includes an item such as a t-shirt, you might want to limit the number of t-shirts that can be purchased simultaneously.
- You can define the minimum and maximum numbers of products that can be owned at any given time. For example, if a product provides an email service, you might want to limit the number of email login names that a single customer can own.

Restricting When a Product Is Available

You can specify that a product is always valid, valid from a future date forward, valid until a future date, or valid for some definite period in the future as shown in Figure 11-8.

Figure 11-8 Restricting Product Availability



For example, if you accept the default value, as shown above, your product is available immediately and always.

You can also specify:

- When each rate is valid. See "Real-Time Rating Based on Date and Time".
- When products and their fees are effective for the accounts that purchase them. See "About Product and Discount Validity Periods".

Specifying Product Priority

When more than one product applies to an event, BRM considers products in the order of product priority. You set product priority when you create a product.

For information on enabling product priority while applying cycle fee, see the discussion about enabling product priority while applying cycle fee in *BRM Configuring and Running Billing*.

Rating Based on Product Provisioning

Product provisioning enables you to define how a service is configured and to rate each configuration accordingly. To activate provisioning, you select a provisioning tag.

For more information, see "About Using Product-Level Provisioning to Configure Services".

Providing Deal-Level Discounts

You use deals to specify discount amounts. You use balance impacts to specify that the resource in the appropriate balance impact is discountable.

> **Note:**
>
> These discounts apply only to events rated by real-time rating. You can create sophisticated discounts that apply to both events rated by real-time rating and events rated by the batch pipeline.

For example, you might have a product that rates two cycle events. For one event you give free hours and for the other you charge a monthly fee. You must create a deal that discounts the monthly fee. However, when you provide a discount in a deal, the discount applies to all cycle forward fees; you cannot choose which cycle fee to discount. To discount only the monthly fee, you make the resource for the monthly fee discountable and the resource used to give free hours non-discountable.

The results of a discount can be calculated two ways. By default, the discount is applied to the product of the rate and quantity:

[(rate * quantity) – discount]

If you prefer, you can apply the discount to the rate itself:

[(rate – discount) * quantity]

See "Setting Optional Rating Flags" for instructions about setting this option.

Defining Extended Attributes

When you create your price list using the XML Pricing Interface, you can define extended attributes for product, deals, and plans. Extended attributes store information used by external applications, such as enterprise product catalogs, beyond what is provided in BRM. See "Defining Product Specification Attributes for Pricing Components" for more information.

About Deals

A deal is a set of products. Deals are typically used for the following purposes:

- To package a set of related products.
- To provide discounts on products for promotional purposes.
- To define start and end dates for a product. The start and end dates that a deal provides must be within the valid start and end dates of the product.
- To allow customers to purchase more than one of the same product with only one transaction.
- To enable on-demand billing.

For example, you can offer two different deals for Internet access:

- A deal that includes a $15 setup fee.
- A deal that has no setup fee.

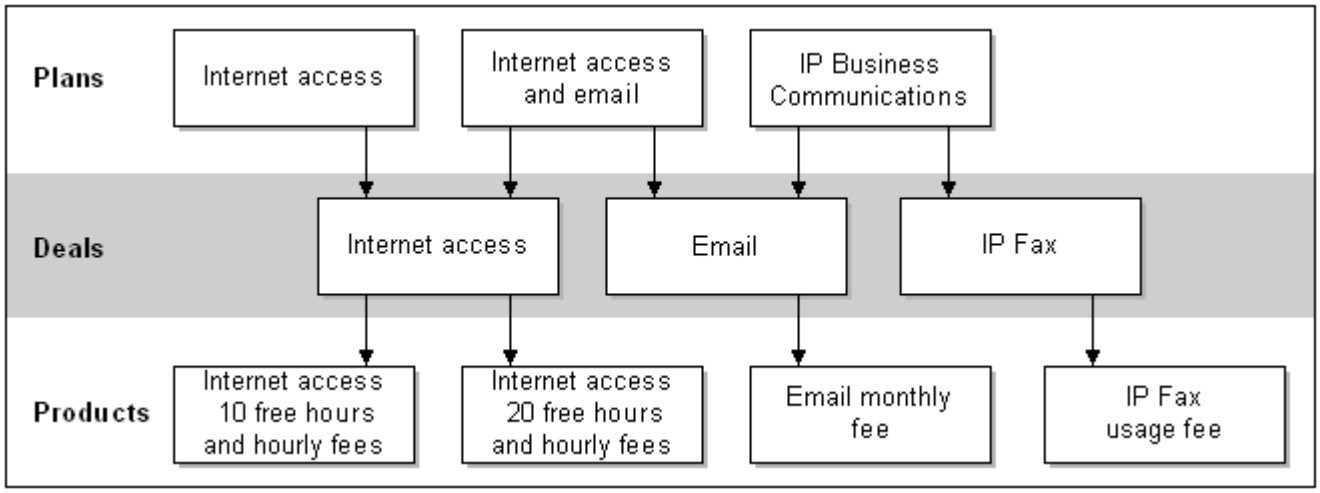
Each deal is typically associated with a specific service. For example:

- A plan that offers only Internet access includes only Internet access deals.
- A plan that offers Internet access and email includes two deals, one for Internet access and one for email.
- A plan that offers email and IP fax service includes two deals, one for email and one for IP fax.

One deal can contain any number of products, and two different deals can contain the same product. Therefore, packaging products in deals adds flexibility to your pricing structure without requiring you to create additional products. Figure 11-9 shows a sample offering consisting of related plans, deals, and products.

As with products, you can define the valid dates for deal availability, and the purchase level of the deal.

Figure 11-9 Deal Relationships to Plans and Products



If you are using the XML Pricing Interface to create your price list, you can also:

- Define what happens if a customer purchases the same product or discount in a deal more than once. By default, it is purchased as a new product or discount, but you can specify that it instead replaces the existing product or discount, or extends the existing product. See "Purchasing the Same Product or Discount Multiple Times" for more information.
- Define how the cycle for products align if a customer suspends the deal and later reactivates it. By default, the cycle aligns with either the billing date or the original purchase date, but you can specify that it instead aligns with the reactivation date. See "Setting Product Cycle Alignment for Reactivated Deals" for more information.
- Define how each product in a deal calculates prorated cycle fees. By default, cycle fees are prorated based on the actual number of days in a month, such as 28, 30, or 31, but you can specify to calculate prorated cycle fees based on a 30-day month, regardless of the number of days in the month. See "Setting Proration for Products in a Deal" for more information.

About the Best Pricing Configuration

Best pricing is a pricing configuration that consists of a base deal and a set of alternate deals that are used to compare charges with the base deal to arrive at the lowest rate for the customer. For more information, see "About Base Deals" and "About Alternate Deals".

Because a deal is associated with a service, best pricing calculation is performed at the service level. You can calculate best pricing for multiple services by grouping services into subscription groups. For more information, see the discussion about how BRM calculates the best price for subscription groups in *BRM Configuring and Running Billing*.

> **Note:**
>
> If the best pricing deal is purchased at the account level, best pricing calculation includes all the events from all the service instances of the account. The products and discounts from all services are included in rating.

About Base Deals

A *base deal* is the deal that is used to rate events as they occur. For each base deal, you can set up a set of alternate deals for a best pricing configuration.

At billing time, the charges calculated using the base deal are compared with the charges of each alternate deal to arrive at the best price.

> **Note:**
>
> You can configure only one base deal in each best pricing configuration.

About Alternate Deals

An *alternate deal* is a deal that is used to rate the events at billing time for comparing the resulting charges with the charges calculated using the base deal. BRM performs a calc-only rerating operation using the alternate deals at billing time. You can also calculate the best price at any time during the billing cycle by calling the PCM_OP_SUBSCRIPTION_CALC_BEST_PRICING opcode in calc-only mode or by using Customer Center. You can then charge your customer the lowest price calculated using the deal that costs the least.

> **Note:**
>
> The number of alternate deals in your configuration affects performance of the best pricing calculation.

For more information, see the discussion about offering the best price to your customers in *BRM Configuring and Running Billing*.

For each alternate deal, you can specify:

- A minimum amount that is charged for the deal whether the services are used or not. For example, the minimum amount can be the sum of the cycle fees applicable for the deal.
- A set of conditions that must be met for the alternate deal to qualify for best pricing calculation.

> **Note:**
>
> The minimum charge and the conditions limit the number of alternate deals that are considered for the best pricing calculation and thereby improve the performance of the best pricing calculation.

An alternate deal can have additional charges that are not in the base deal, and the rates in the alternate deal override the rates in the base deal.

An Example of Best Pricing Calculation

Consider an account that has used a total of 700 minutes and 10 Short Message Service (SMS) messages in the month and has the best pricing configuration shown in Table 11-7:

Table 11-7 Best Pricing Calculation Example

Base Deal	Alternate Deal
Minutes usage charge = $0.05	Minutes usage charge = $0.03
SMS usage = $0.10	Free SMS usage
NA	Cycle fees = $10

The charges calculated are shown in Table 11-8:

Table 11-8 Calculating Charges

Usage Type	Base Deal Charges	Alternate Deal Charges
Minutes	700 x 0.05 = $35.00	700 x 0.03 = $21.00
SMS	10 x 0.10 = $1.00	0
Cycle fee	0	$10
Total	**35 + 1 + 0 = $36**	**21 + 0 +10 = $31**

In this example, the alternate deal charges are lower than the base deal charges, so the alternate deal is the best deal and is used for rating and billing the customer.

Purchasing Deals after an Account Is Created

When a customer registers for an account, the customer purchases deals contained in a plan. However, once an account is created, the customer can purchase add-on deals that are not included in a plan, but the deal must be associated with the same service.

Providing Discounts with Deals

You use deals to provide discounts on products. You can provide separate discounts for each type of rate. For example, if the product contains a cycle rate (for a monthly fee) and a usage rate, you can discount either or both rates.

For each rate category in each product, you can specify a percentage to discount and the dates the discount applies to.

When configuring discounts, you can:

- Specify if rates are discountable. See "Providing Deal-Level Discounts".
- Provide fractional discounts up to two decimal places (for example, 10.25%, but not 10.255%).
- Define what happens if a customer purchases the same discount more than once, if you are using the XML pricing interface to create your price list. By default, the new purchase is treated as a new, separate discount, but you can specify that it instead replaces the existing discount. See "Purchasing the Same Product or Discount Multiple Times" for more information.

Prohibiting, Allowing, and Requiring Deal Modification

You can also give discounts to products in Customer Center (for example, fixed or percentage discounts on monthly fees). When you create a deal, you can specify whether to prohibit, allow, or require deal modification.

For example, to set a price based on individual customer input, you can specify that a deal must be modified.

About Product and Discount Validity Periods

Products and discounts have the following validity periods:

- The validity period that defines when a product or discount is generally available for purchase. You define this validity period when creating products and discounts. See "Restricting When a Product Is Available".
- The validity periods that define when the product and discount are effective for the accounts that purchase them and when the product's fees begin to accrue in the account balance and to be discounted. You define these validity periods when adding products and discounts to deals. See "Setting the Effective Periods of Products and Discounts".

Setting the Effective Periods of Products and Discounts

You specify the effective period of products and discounts by defining when the purchase, cycle, and usage periods start and end:

- The purchase period defines when the product or discount is activated and the customer can begin to use the product's service or benefit from the discount. The product's purchase start time is also the earliest time that the product's fees can begin to accumulate in the account balance.
- The cycle period defines when the cycle fees are charged or discounted.
- The usage period defines when the usage fees are charged or discounted.

The cycle and usage periods must fall within the purchase period.

You can set the purchase, cycle, and usage periods to start:

- **Immediately**: The effective period starts as soon as the customer purchases the product or discount.
- **On first usage**: The effective period starts when the product or discount is first used. The product fees are not charged until the customer uses a service in the product for the first time: for example, by making a phone call. For more information, see "About Effective Periods That Start on First Usage" for more information.
- **Relative to the product or discount purchase time**: The purchase time is the time when a product or discount is added to the account.
  - When the purchase period has a relative start time, the purchase fee is charged when the product is purchased, but the customer cannot use the product's service or benefit from the discount until the relative period ends.
  - When a product's cycle or usage period has a relative start time, the cycle or usage fees are not charged (or rated) until the relative period ends, even if the service has been activated. This option enables you to waive subscription or usage fees for a period.
  - When a discount's cycle or usage period has a relative start time, the discount is not applied to cycle or usage fees until the relative period ends.
- **On a specific day of the month**: If you use the XML pricing interface to create your price lists, you can also align purchase, cycle, and usage periods to a specific day of the month. See "Aligning Recurring Charges and Product Validity to a Specific Day of the Month" for more information.

You can set the purchase, cycle, and usage periods to end:

- **Never**: Once activated, the product or discount is effective indefinitely. The product's fees can be charged or discounted indefinitely.
- **Relative to the start time**: The start time is when the product or discount is activated.
  - When the purchase period ends relative to the purchase start time, the product or discount is effective for the relative period specified. After the relative period ends, the customer can no longer use the product's service or benefit from the discount.
  - When a product's cycle or usage period has a relative end time, the cycle or usage fees are no longer charged when the relative period ends.
  - When a discount's cycle or usage period has a relative end time, the cycle or usage fees are no longer discounted when the relative period ends.

About Effective Periods That Start on First Usage

Setting products and discounts to start when they are first used enables you to delay charging customers for the services they purchase until they start using those services or to delay activating discounts until they can be applied to customers' usage. This is useful when you offer limited-time services or discounts that expire relative to when they are activated.

> **Note:**
>
> Products that start on first usage must include usage fees. If you set a product that has no usage fee to start on first usage, the product will never be activated.

You can also set up individual resources granted by products and discounts to start when the resource balances are first consumed. For more information, see "About Balance Impacts That Become Valid on First Usage".

The effective period start time is set to the start time of the event that first uses the service or triggers the discount. The end time is set based on the end time that you configure for the product or discount.

> **Note:**
>
> If you use the XML pricing interface to create your price lists, you can specify that all products and discounts in a deal are activated when one of the products or deals is activated.
>
> For example, assume a deal includes products A and B and discount C. If the first service a customer uses is in product B on 15 May, all three (products A and B and discount C) are activated on 15 May. See "Activating Products in Plans and Deals on First Usage" for information and examples.

About Activating First-Usage Discounts

Discounts that start on first usage can be activated when the customer first uses a service, when the first cycle fee is applied, or when the account's bill is generated, depending on which fees are discounted and when the discount is triggered. For example, a first-usage discount on SMS messaging is activated when the customer sends the first SMS message, a first-usage discount on cycle fees is activated when the first cycle fee is applied, and a first-usage billing-time discount is activated when the first bill is generated for the account.

A discount on a particular service may not be activated when a customer first uses that service. For example, if long-distance calls are discountable for a telephony service, a customer may make multiple local calls, which do not trigger the discount's effective period, before making a long-distance call.

There are some cases in which a discount's balance impact is negated by a second discount. If this occurs, the first discount's validity period is still set.

For example, a customer purchases a plan that includes discount A and discount B:

- Discount A is configured to start when first used, gives 10% off of all calls, and has a higher priority, so it is applied first.
- Discount B is configured to start immediately and makes all birthday calls free.

If a customer makes a first-usage call on his birthday and is charged $5.00 for the call, discount A is applied first, which reduces the balance by $.50, and discount A's validity period is set. Then discount B backs out all charges. In this case, the validity period of discount A remains set.

About Setting First-Usage Validity during Pipeline Rating

If you use Pipeline Manager to rate usage, configure these pipelines to set the effective periods of products and discounts when they are first used:

- Batch rating pipeline. See "Configuring Pipeline Output for First-Usage Products, Discounts, and Balance Elements".

- Batch discounting pipeline. See "About Setting the Validity of Balance Elements Impacted by Discounts ".
- Real-time rerating pipeline. See "Configuring the Real-Time Rerating Pipeline to Set Charge Offer Validity Periods".

About Product and Discount Status at Purchase

When you add a product or discount to a deal, you also specify whether the product or discount is active or inactive at the time of purchase. Additionally, for products or discounts with inactive status, you must also specify a reason code that further describes the reason for the inactive status.

You define reason codes in the **reasons.***locale* file. For more information, see "Localizing and Customizing Strings" in *BRM Developer's Guide*.

Assigning Services to Deals

You define the services that a deal applies to by assigning a purchase level to a deal. The purchase level can be any of the following:

- **A single service**. Usually, you set a deal's purchase level to a single service. The products in the deal and the rates they contain can only be applied to that service. This makes it easier to create plans: you have more flexibility in putting together a combination of deals when each deal applies to a specific service. Also, when a CSR adds a service to an account, it is easy to find the correct deals to offer when the deals are associated with individual services.
- **All accounts/no services**. When you set the purchase level to all accounts/no services, you create an *account-level deal*. Each account can have only one account-level deal. The account-level deal typically includes account-level products (for example, a product that specifies the monthly charge for owning an account, regardless of the services).

Using Deals to Bill Customers on Demand

On-demand billing enables you to bill a customer immediately for a purchase, even if the customer's billing cycle has not ended.

When you create a deal, you can flag it for on-demand billing. When a customer purchases a deal that is flagged for on-demand billing, a bill is generated immediately for the purchase fees associated with the deal.

> **Note:**
>
> On-demand billing works with purchase fees only, not with cycle, usage, or cancel fees.

For more information about on-demand billing, see the discussion about on-demand billing in *BRM Configuring and Running Billing*.

About Deal Dependencies

You can define dependencies between deals that set up the following relationships:

- **Prerequisites**. Specifies that an account must own a particular deal to be able to purchase an additional deal.

  > **Note:**
  >
  > – A prerequisite can contain deals of different services. For example, to own a GPRS deal, an account must own a GSM deal.
  >
  > – A prerequisite deal cannot contain item products. When you create a plan with alternate deals, and if the base deal contains an item product, the purchase fails.

- **Plan requirements**. Specifies whether deals are optional or required for plans. Required deals must be purchased when a plan is purchased, whereas optional deals can be added at any time.
- **Mutual exclusivity**. Sets up a mutually exclusive relationship between two deals so if an account owns one deal, it cannot own the other.
- **Allowed transitions**. Specifies which deals or plans can serve as replacements for others.

  Transitions specify the deals that customers can switch to and remain fully provisioned. While transitioning from one deal to another, your customers retain their devices, such as phone numbers and services.

  Customers owning deals associated with a primary service may transition to other deals associated with that primary service. The list of deals displayed as available for transition are all associated with a specific primary service.

Strategies for Creating Deals

You typically use deals for providing discounts. This enables you to create fewer products and to create a baseline of standard products that you can manipulate with deals.

For example, you could create several products, each with different rates. Or, you could create a single product and change the rates by providing discounts in deals.

Also, to make your deals easier to handle in Customer Center, you should associate each deal with a single service. A common exception is a deal that applies to all accounts, which you associate with accounts instead of a service.

When you create a deal, you can specify whether to prohibit, allow, or require deal modification. In addition, when you create a plan with options, you can specify whether the deals contained in the plan are required or optional when the plan is purchased.

Using Deals across Time Zones

When you create a deal, BRM sets the deal's start and end dates to the current date at midnight GMT (Greenwich Mean Time). For example, if a user in the US creates a deal at any time on January 15, the system stores it as January 15 at midnight GMT. If the BRM server is located in London, the deal's time is set to GMT London time, 5 hours later than the US-based user's computer. If the US-based user opens a deal using **Modify Product**, the deal's start time is posted as 5 hours earlier than midnight on January 15 (or 7:00 PM on January 14). The start date is then recorded as January 14 rather than January 15.

To avoid a time discrepancy, add the following string to the **Infranet.properties** file:

```
infranet.user.timezone = time_zone
```

where *time_zone* is the appropriate time zone, such as **Europe/London** or **America/Los_Angeles**.

Transitioning between Deals

You can perform three types of transitions between deals: upgrade, downgrade, and generation change. You define the deals that can be transitioned when creating deals in Pricing Center.

For generation change, you can transition customers between 2G (second generation) and 3G (third generation) wireless deals and services. Deals are called 2G or 3G depending on whether they have a second- or third-generation service as their primary service type. For more information about defining generation change transitions, see "Defining a Generation Change for Packages" in *BRM Managing Customers*.

> **Note:**
>
> For service-level extended rating attributes (ERAs), be aware that ERA profile information is not automatically transferred between deals during deal transition or generation change. If the two deals have some common provisioning tags, the ERA profile information can be reconfigured in the new deal.

If you create pricing plans using XML and the **loadpricelist** utility, you can also define how to apply cycle rates when customers transition from one deal to another in the middle of their billing cycle. By default, the cycle rates of both deals are prorated, but you can specify instead to apply only the original deal's cycle rate or only the new deal's cycle rate. See "Transitioning Plans and Deals " for more information.

About Add-On Products in Deals

When you create your price list using the XML Pricing Interface, you can include add-on products in your deals.

All products in deals are base products by default, which means they can be purchased without any prerequisites. Add-on products can be purchased only if the customer owns a valid base product.

When you create an add-on product, you also specify how to determine its validity start date. The add-on product's validity start date is the end date of a product that you specify. For example, assume product A has a validity period from June 1 through June 15. If you specify to align add-on product B's validity period with product A, product B's validity start date would be June 15. See "Configuring Add-On Products in Deals" for information.

Defining Extended Attributes

When you create your price list using the XML Pricing Interface, you can define extended attributes for product, deals, and plans. Extended attributes store information used by external applications, such as enterprise product catalogs, beyond what is provided in BRM. See "Defining Product Specification Attributes for Pricing Components" for more information.

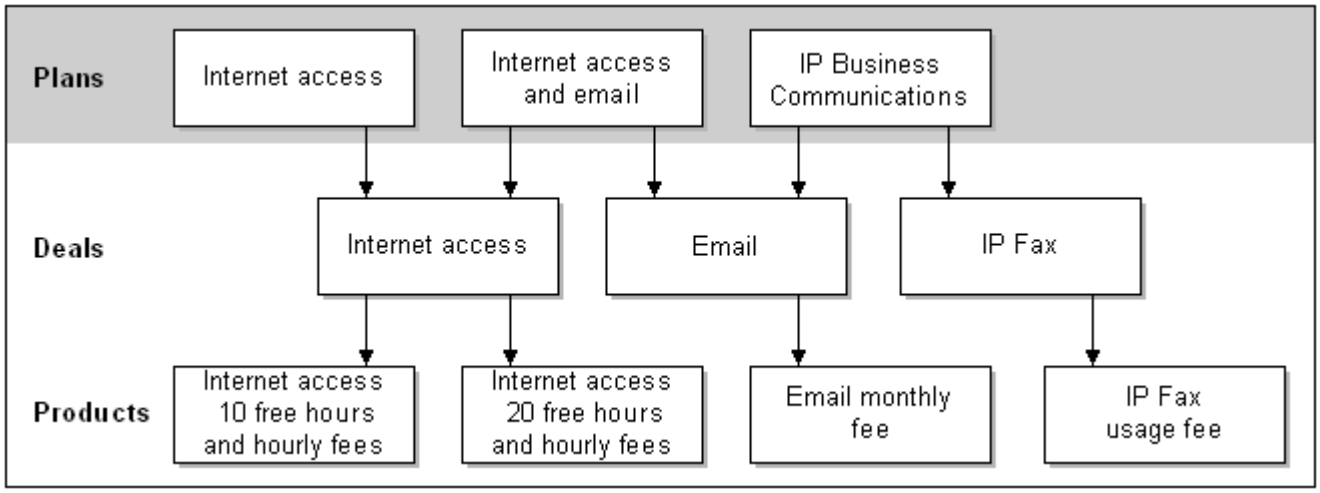
About Plans

You use *plans* to offer your services to customers. For example, if your company sells Internet access, email, and IP fax service, your plans might include:

- A plan that sells only Internet access.
- A plan that sells Internet access and email.
- A plan that sells email and IP fax service.

Figure 11-10 shows how you can include deals, and the services they apply to, in a variety of plans.

Figure 11-10 Plan Relationships to Deals and Products



Usually, plans include services. By including services in plans, a customer creates a service login name and password during registration. The only time you might not include a service in a plan is if the plan contains only account-level products and deals. In that case, you register a customer, and assume that services are added later.

One plan can contain any number of deals, and two different plans can share the same deal. By grouping deals into plans, you simplify the choices presented to customers.

> **Note:**
>
> Plans always include deals. The only exception is the CSR plan.

About Applying Credit Limits to Resources

You use plans to apply credit limits to resources.

A *credit limit* is the maximum amount of a resource, such as currency or hours, that can accumulate in an account balance before the customer is prevented from logging in or using a service. For example, you might set a credit limit of $100 for an Internet access plan.

> **Note:**
>
> The default credit limit for currency resource is NULL and noncurrency resource is 0. The default currency resource must have a positive value and the noncurrency resource must have a negative value.

Credit limits are defined in the plans that customers purchase. Credit limits must also be defined in the input flist when the customer account is created.

- If the credit limit is not defined in the input flist, the credit limit defaults to NULL even if the credit limit is defined in the plan.
- If the credit limit defined in the plan differs from the credit limit in the input flist, the credit limit in the input flist is used.

You use Customer Center to change the credit limit.

Though the credit limits are defined in the plans that customers purchase, the same credit limits must be defined in the input flist when the customer account is created.

About Credit Thresholds and Credit Floors

You set credit thresholds to notify customers when they are approaching the credit limit of a resource. Credit thresholds are defined in plans.

The *credit threshold* specifies the balance total that triggers an alert to the customer. You can specify the threshold in two ways:

- As a fixed value, such as $100 or 30 minutes.
- As a percentage of the credit limit, such as 90%. For example, if the credit limit is $100 and the threshold is 90%, the threshold amount is reached when the customer has a balance of $90; that is, when the customer has used 90% of the resource.

The *credit floor* is the starting point for the credit thresholds and is the lowest number that the resource value can be; that is, the number that represents no use of the resource. For currency resources, the credit floor is 0.

For noncurrency resources, such as prepaid hours, you can use a negative number for the credit floor. For example, if you give 100 prepaid hours and you set the credit limit to **0**; when the credit limit is reached, the customer has no hours remaining and cannot use the service.

You can also set the plan to use a dynamic credit floor. In this case, the credit floor is determined by the granted amounts from the sub-balances that are valid for the current cycle for the resource (for example, minutes).

The threshold works the same way, regardless of whether the credit floor is fixed or dynamic.

To notify the customer when there are only 10 hours left, you set the credit threshold and floor as follows:

- Set the credit floor to -100. This is the number that indicates none of the resource has been used.
- Set the credit threshold to 90%.

The threshold is reached at 90% of -100 hours; that is, when the customer has 10 prepaid hours left.

> **Note:**
>
> You can adjust a customer's credit limit.

The credit threshold is triggered both when the balance increases and when it decreases.

You can enable BRM to provide credit threshold breach notifications as in-session notifications appended to the responses it sends to network connectivity applications. BRM sends these notifications in its responses to the authorization and reauthorization requests that BRM receives from these applications during prepaid sessions. For more information on in-session notifications in prepaid sessions, see the discussion about providing in-session notifications for network connectivity applications in *BRM Telco Integration*.

You can customize BRM to perform different actions in each case. For example, if the credit threshold is crossed when the balance is increasing, service could be turned off. When the threshold is crossed when the balance is decreasing, service could be restored.

Credit Limit and Floor Options

There are two optional settings that affect the way credit limits and floors are handled during rating:

- You can choose to turn off the checking of credit floors. Credit floor checking sometimes causes certain events to be skipped during rating.
- You can choose whether to impose a zero credit limit if there is a balance impact to a resource in an account where the balance element for that resource is missing. If you choose to use a zero credit limit, the balance for that resource cannot exceed zero. If you choose not to use the credit limit, the balance can be any amount.

You set these options by changing entries in the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**). See "Setting Optional Rating Flags" for details.

Setting a Dynamic Credit Floor

Instead of setting a static credit floor, you can set BRM to generate a dynamic credit floor from the granted amounts from the sub-balances that are valid for the current cycle for the resource (for example, minutes).

You can set a dynamic credit floor either by using the **loadpricelist** utility or by using the PCM_OP_PRICE_SET_PRICE_LIST opcode. You can also set the plan to use a dynamic credit floor using the PCM_OP_BILL_SET_LIMIT_AND_CR opcode. See "How BRM Handles Consumption Rules and Credit Limits" for more information about using PCM_OP_BILL_SET_LIMIT_AND_CR.

To set a dynamic credit floor using the **loadpricelist** utility, include the **dynamic_floor** tag, with a value of **1**, in the XML file processed by loadpricelist.

To set a dynamic credit floor using the PCM_OP_PRICE_SET_PRICE_LIST opcode, include the PIN_FLD_DYNAMIC_CREDIT_FLOOR field in the PIN_FLD_LIMIT array and set the value to **1**.

Regardless of the method used, if the dynamic credit floor is not set, the default value of **0** will be used.

See "Enabling Dynamic Credit Floors in Plans" for more information about setting this field.

About Effective Periods That Start on First Usage

Setting products and discounts to start when they are first used enables you to delay charging customers for the services they purchase until they start using those services or to delay activating discounts until they can be applied to customers' usage. This is useful when you offer limited-time services or discounts that expire relative to when they are activated.

> **Note:**
>
> Products that start on first usage must include usage fees. If you set a product that has no usage fee to start on first usage, the product will never be activated.

If you use the XML pricing interface to create your price lists, you can set all products and discounts in a plan to be activated when one of the products or deals is activated.

For example, assume a plan includes deal 1 with products A and B, and deal 2 with product C and discount D. If the first service a customer uses is in product B on 15 May, all four (products A, B, and C, and discount D) are activated on 15 May. See "Activating Products in Plans and Deals on First Usage" for information and examples.

Tracking Resources by Service

Accounts can have multiple balance groups. By default, accounts are created with one balance group. You can add additional balance groups to track resources for specific services when you create your plans in Pricing Center.

For example, to track resources for a $50.00 prepaid wireless service, you create a balance group for the service in the plan and specify a credit limit of $0.00. You specify $0.00 because prepaid services have a credit balance, represented by a negative number, in this case, -$50.00. As customers use the service, the balance is debited until it reaches $0.00.

If you do not create additional balance groups, resources for every service a customer purchases share the same balance group. This means that resources such as free minutes are shared among all services. By creating a balance group for each service, you can control the allocation and consumption of resources for each service instance.

When customers purchase plans, the associated balance groups are added to the customers' accounts. Customers' account balances are then displayed in Customer Center for each service or set of services stored in a balance group.

CSRs can also specify credit limits for the services in a balance group when they create customer accounts. For example, customers might want to limit the monthly amount they spend on phone calls. CSRs set credit limits in Customer Center.

Grouping Services by Subscription

You can group services by subscription (for example, a set of services associated with a wireless connection). You group subscription services to track balances and bill customers for individual subscriptions rather than for the accounts.

An account can contain multiple subscriptions. To group subscription services, you define the service that represents the subscription, then associate it with the services that customers actually use. For example, you might use a telco service to represent the subscription and

associate it with telephony and messaging services. When customers purchase the telco subscription and use the associated services, the service fees are tracked and stored at the subscription level.

Creating CSR Plans

In addition to providing services to your customers, plans provide services to CSRs.

To create a CSR plan:

1. Use Pricing Center to create a plan that has the following attributes:
   - Use the **admin_client** service. This service provides access to Customer Center users.
   - Include no deals.
2. Add the CSR plan to the **CSR - new** plan list.

CSR plans serve two purposes:

- They control access to Customer Center.
- They allow customer management events to be recorded, including information about the CSR who generated the event. This information can be helpful when researching customer complaints.

Using Plans to Bill Customers on Demand

On-demand billing enables you to bill a customer immediately for a purchase, even if the customer's billing cycle has not ended.

When you create a plan, you can flag it for on-demand billing. When a customer purchases a plan that is flagged for on-demand billing, a bill is generated immediately for the purchase fees associated with the plan.

> **Note:**
>
> On-demand billing works with purchase fees only, not with cycle, usage, or cancel fees.

For more information about on-demand billing, see the discussion about on-demand billing in *BRM Configuring and Running Billing*.

Strategies for Creating Plans

When your customers register for accounts, they are presented with a list of your plans. Similarly, CSRs see a list of plans when creating accounts. Therefore, you should be careful about your plan names and descriptions.

> **Note:**
>
> A plan name can include a maximum of 255 characters. A plan description can include a maximum of 1023 characters.

You might consider using optional deals when creating plans. This enables you to include all relevant deals in one plan, without requiring customers to purchase all of them. Optional deals are available for purchase at registration time or any time afterward. For example, say a company offers a GSM service plan that includes a standard GSM deal, voice mail deal, and a text messaging deal. CSRs are required to purchase the base GSM deal for the service and have the option of adding the voice mail deal and text message deal for the customer at a later time. Adding optional deals to plans also filters the deal list in Customer Center, so that only the deals within the plan are available when CSRs add on deals to a current account.

Another strategy is to consider creating plans that form a logical upgrade path. For example, one plan might offer a per page rate for IP Fax service, while the higher-grade plan offers unlimited IP Fax service.

A third strategy is to create plan-to-plan upgrades, downgrades, or generation changes. This enables you to easily transition customers from one plan to the next by defining rules that govern how plans can be purchased.

Transitioning between Plans

You can perform three types of transitions between plans: *upgrade*, *downgrade*, and generation change. You define the plans that can be transitioned when creating plans in Pricing Center.

For generation change, you can transition customers between 2G (second generation) and 3G (third generation) wireless plans and services. Plans are called 2G or 3G depending on whether they have a second- or third-generation service as their primary service type. For more information about defining generation change transitions, see "Defining a Generation Change for Packages" in *BRM Managing Customers*.

> **Note:**
>
> For service-level extended rating attributes (ERAs), be aware that ERA profile information is not automatically transferred between plans during plan transition or generation change. If the two plans have some common provisioning tags, the ERA profile information can be reconfigured in the new plan.

If you create pricing plans using XML and the **loadpricelist** utility, you can also define how to apply cycle rates when customers transition from one plan to another in the middle of their billing cycle. By default, the cycle rates of both plans are prorated, but you can specify instead to apply only the original plan's cycle rate or only the new plan's cycle rate. See "Transitioning Plans and Deals " for more information.

Defining Extended Attributes

When you create your price list using the XML Pricing Interface, you can define extended attributes for product, deals, and plans. Extended attributes store information used by external

applications, such as enterprise product catalogs, beyond what is provided in BRM. See "Defining Product Specification Attributes for Pricing Components" for more information.

About Plan Lists

A *plan list* is a group of plans, usually offered to a single type of customer. You use plan lists to group rate plans based on customer types, such as the customer's age and address. For example, you might have the following plan lists:

- A plan list that includes plans for customers above a certain age.
- A plan list that includes plans for customers in a particular location (for example, Canadian customers).
- A plan list that includes promotional discounts that you offer for a limited time.

Grouping plans into plan lists enables you to:

- **Offer plans to customers based on customer type**. For example, in Pricing Center you can define a Gold - Senior plan list with two plans, GS1 and GS2. Based on the age group of the user, you can use the Gold - Senior plan list to limit the offerings to your customers through your customer management application to GS1 and GS2.

  > **Note:**
  >
  > You must maintain mapping between your plan lists and your customer types.

- **Control the rollover of free resources**. For example, you can control rollovers with a rule specifying that when a customer changes plans, free resources can be rolled over only if plans are within the same plan list.

The name and type together identify a unique plan list. The name and type are case sensitive. For example, Gold - Senior and gold - senior are two different plan lists. You can assign any name and type to the plan lists.

You can create any number of plan lists for your database, and each plan list can contain any number of plans. Two different plan lists can contain the same plan.

The plan list does not have to include all of your plans. You can create plans and not include them in a plan list until you need them. Or, you can offer one set of plans to one group of potential customers, and another set of plans to another group.

BRM includes two types of plan lists:

- Use *new* plan lists to register new customers.
- Use *add-on* plan lists to add services to existing accounts.

You can also add your custom plan list based on the customer type.

The plan lists you create are displayed in Customer Center. If you use your own custom application, use the PCM_OP_CUST_POL_GET_PLANS policy opcode to retrieve and display plan lists.

About the Default and CSR Plan Lists

BRM includes special plan lists that you can use to determine where the plans are displayed. For example, you might want some plans to be displayed only in Customer Center.

- Plans in the **CSR - new** and **CSR - addon** plan lists are displayed only in Customer Center.

> **Note:**
>
> The **webclient** name is the default but can be changed. See the discussion about specifying which plans to display in *BRM Managing Customers*.

- Plans contained in the **default-new** and **default-addon** plan lists are used in Customer Center if the **CSR** plan list is not available.

About Offer Profiles

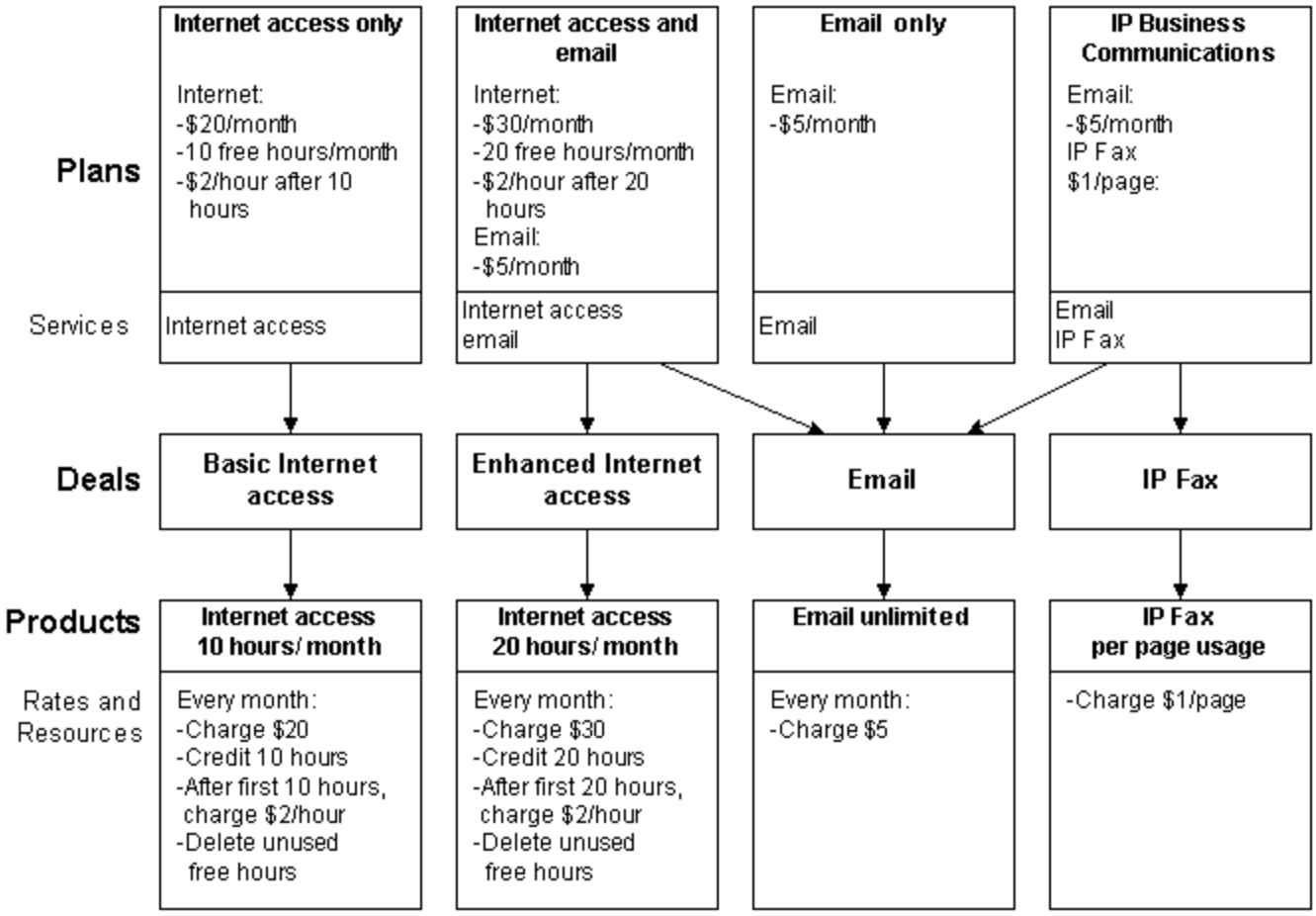
Offer profiles are made up of one or more policy labels each of which defines a gradation in the quality of service (QoS) based on usage amounts for a resource.

For example, you can have an offer profile called *"Platinum"* for a data service and define its resource as *Megabytes Used*. You can define a policy labeled *Fair Usage*, which has three levels, *Low QoS*, *Medium QoS*, and *High QoS*, with each level containing a usage range valid for that quality of service.

Price List Example

In the sample price list shown in Figure 11-11, three services are offered: Internet access, email, and IP fax service. All rate plans used in this price list are real-time rate plans.

Figure 11-11 Sample Price List



Note the following points about how this price list is organized:

- There are two products that define different monthly fees for Internet access. These products are used by different deals.
- Two of the plans include two deals, the other two plans include only one deal each.

For information about the sample plans, see "Understanding the Sample Pricing Plans".

About Setting Up a Price List

Setting up a price list typically requires the following steps:

1. Marketing and finance personnel define which services your company offers, and how much to charge for them. For example, they define how much to charge each month, how much to charge for online usage, and how much to charge to sign up for a plan.
2. Operations personnel review the pricing model to determine if any new services must be created to support the price list. In addition, they determine if any new resources, services, events, general ledger (G/L) IDs, or tax codes must be created. See "Prerequisites for Creating a Price List".
3. An operations person plans the price list (for example, to determine how many plans and deals to create and to determine the products that make up the plans). This plan is

typically reviewed by marketing and finance personnel to confirm that it implements the company's pricing model. See "Planning Your Price List".

4. An operations person uses Pricing Center to create the price list. Because products are the basic unit of the price list, they are created first. See "About Using Pricing Center".
5. An operations person tests the price list. See "Testing Your Price List".

Prerequisites for Creating a Price List

Before you create a price list, you must complete the following tasks:

- **Create services and events**

  BRM includes Internet access and email services by default, but you might need to modify them or add new services before you create your price list. You must also configure a list of events to track for each service. If you create new services, you may need to create new events to track them.

- **Create resources**

  Rate plans require resources. Use the Resource Editor in Pricing Center to add or modify resources. For more information, see "About Resources".

- **Define currency conversion rates**

  Using more than one currency requires that you define currency conversion rates.

- **Create G/L IDs**

  You use G/L IDs to collect general ledger information from the BRM database and export it to your accounting application. You must decide how to track the revenue for each type of rate and create the appropriate G/L IDs.

- **Define tax codes and tax suppliers**

  To calculate taxes, you must define tax codes and tax suppliers.

- **Set up offer profiles**

  To ensure that resource tracking for policy-driven provisioning of services, use the offer profile name as the provisioning tag in the product or discount. Set the resource used in the offer profile as the resource counter in the product or discount.

- **Define product-level provisioning categories**

  If you use product-level provisioning, you must define provisioning tags. See "About Using Product-Level Provisioning to Configure Services" and "Working with Provisioning Tags ".

- **Define ratable usage metrics (RUMs) for events**

  You use RUMs to identify the event attributes to rate for each event. RUM definitions are stored in the BRM database. You use a text editor and a utility to define RUMs. For more information, see "About Ratable Usage Metrics".

- **Map event types to services**

  When you create a product, you select a service and events you want to rate that are related to that service. Because all event types are not valid for all services, you map event types to services. Creating this map prevents you from selecting an event that does not occur for a given service. See "Mapping Events and Services".

- **Define zones**

  For real-time rating, you use zones to create a single value to represent a group of values. You use the representative value in a rate plan selector. See "About Real-Time Zoning".

For information on zones in batch pipeline rating, see "Setting Up Zones for Batch Pipeline Rating".

- **Define impact categories**

  For real-time rating, you use impact categories to specify that a particular group of balance impacts within a rate should be used. If you plan to use attribute value grouping during rating, you must define some impact categories. See "Real-Time Rating Using Event Attributes".

  For pipeline rating impact category information, see "About Impact Categories".

- **Define pipeline data**

  If you use pipeline rating, you must define several types of data and pricing components. See "Setting Up Real-Time and Pipeline Pricing and Rating".

Planning Your Price List

When you plan your price list, you determine how much to charge for your services, for example:

- Which types of fees to use for a service, such as a flat monthly fee, hourly usage fees, or both.
- The rates to charge for monthly fees, hourly usage, set up, and so forth.
- Discounts such as those based on time of day, date of purchase, or volume usage.
- How to handle multiple currencies.
- Special pricing options such as sponsored rates.

> **Note:**
>
> Because price lists are built by setting up relationships among plans, deals, and products, it is helpful to draw a diagram of your price list before you create it in Pricing Center. For an example, see "Price List Example".

About Using Pricing Center

You use Pricing Center to create and modify your price list.

> **Note:**
>
> Offer profiles cannot be created or modified using Pricing Center. Use the **loadpricelist** utility or the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE opcode.

With Pricing Center, you can perform the following tasks:

- Create and modify products, deals, plans, and plan lists.
- Create and modify rate plans for products. You can create rate plans for real-time rating and for pipeline batch rating.

- Define complex rate plans for your services based on valid time periods, event quantities, and rate priorities.
- Set credit limits and discounts.
- Reuse deals, plans, products, and rate plans so you do not need to re-create them every time.
- Update your pricing model without stopping and restarting your BRM system.
- Save a pricing document to a binary file and open the file at a later date. This enables multiple people to work on different pricing objects that will be committed to the same database.

Who Uses Pricing Center?

Generally, the services you offer and how you charge for them are defined by your marketing and finance personnel. The price list is typically created by operations personnel using Pricing Center. You should outline, in detail, your entire pricing structure before you use Pricing Center to implement it.

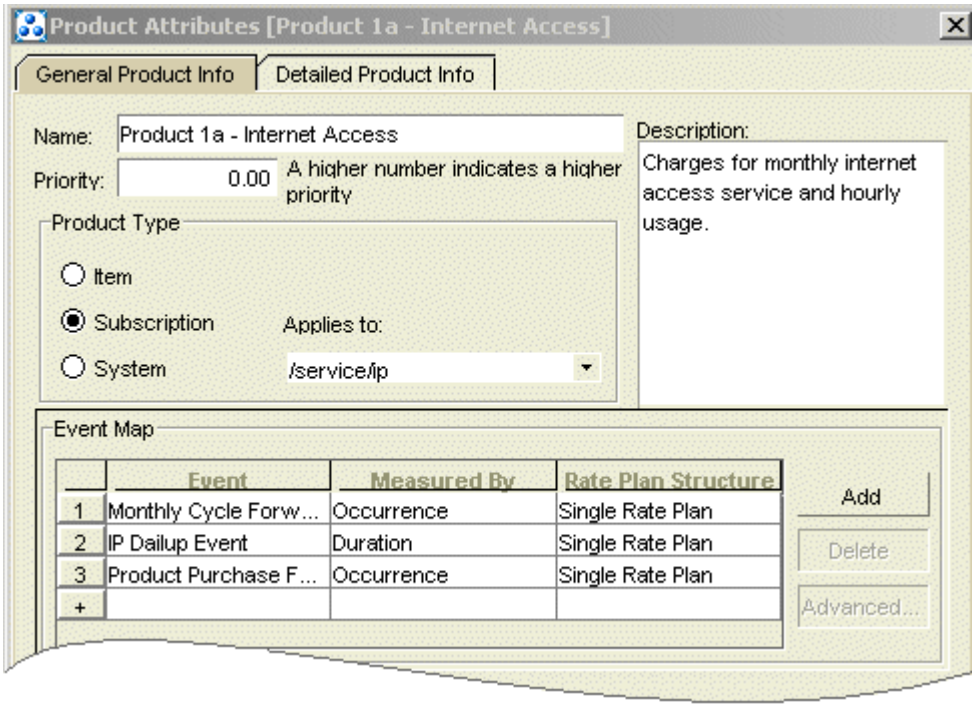
Example of Using Pricing Center

When you are ready to use Pricing Center to create a price list, the price list should be planned, and all necessary components, such as resources, G/L IDs, and RUMs should be in place. (See "Prerequisites for Creating a Price List" and "Planning Your Price List".)

You perform the following steps to create a price list by using Pricing Center:

1. Start by using the Product Creation wizard to create the products. The wizard steps you through the general product parameters, such as the name and description of the product, the service it applies to, and the start and end dates.
2. When you finish, Pricing Center displays the product attributes that you selected. You can change them at any time.
3. After the general product properties are defined, you define the rate plans that specify how much to charge. In the example illustrated in Figure 11-12, the product includes an IP connection fee, a monthly fee, and a purchase fee, so you create rate plans for those events:

Figure 11-12 Product Example

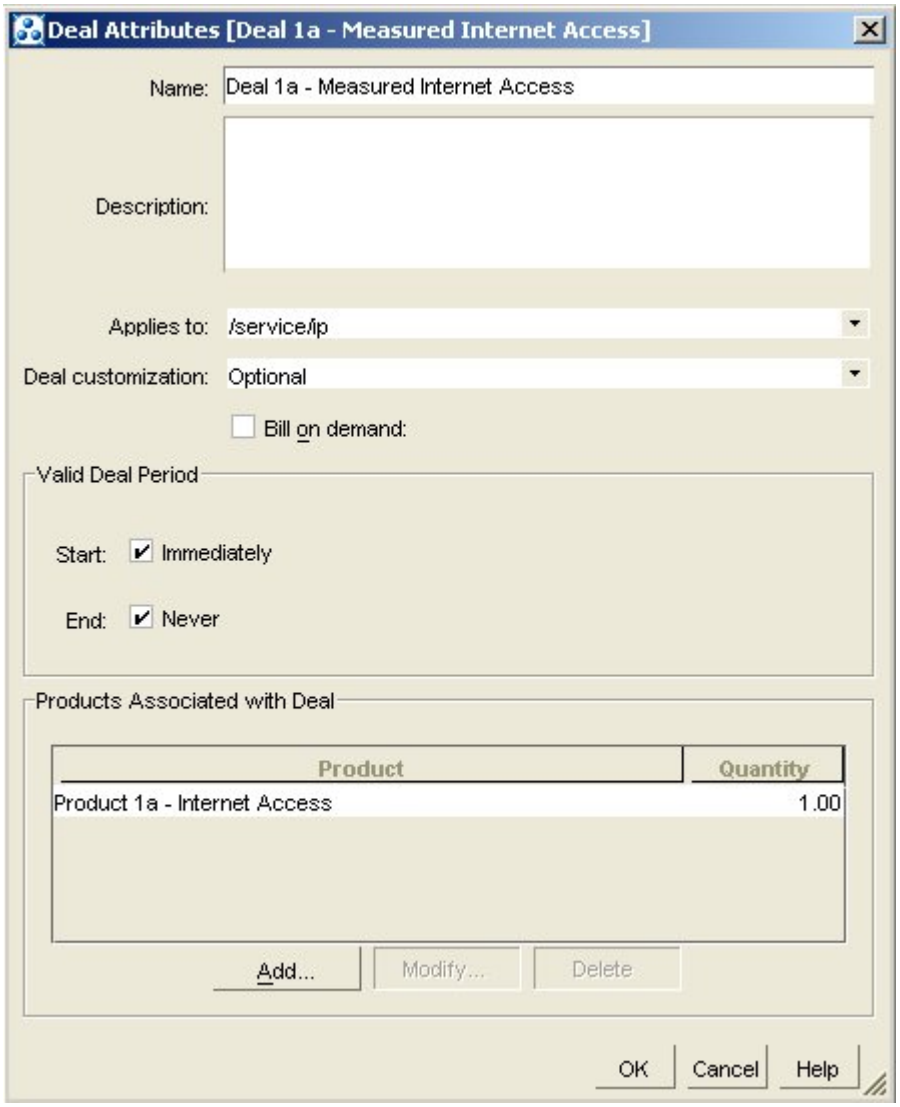


4. For real-time rate plans, you apply balance impacts to each rate plan. Figure 11-13 shows the balance impacts for a purchase fee:

Figure 11-13 Balance Impact Example

5. When the product is finished, you create a deal and add the product as shown in Figure 11-14:

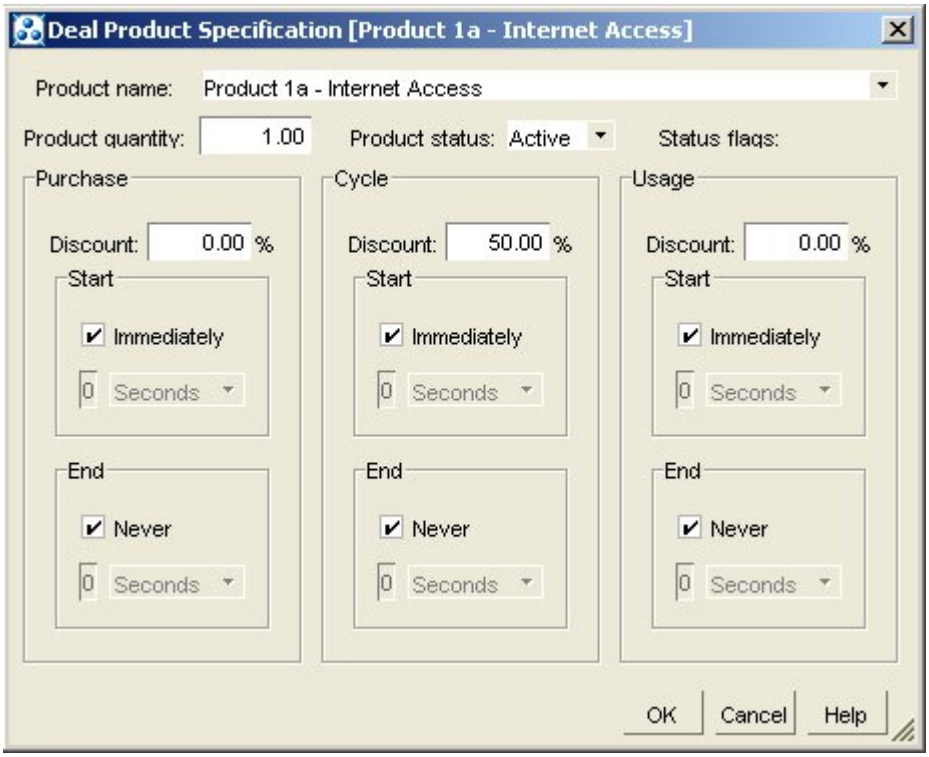
Figure 11-14 Adding Product to Deal Example



6. Then you tailor the product settings specifically for the deal.

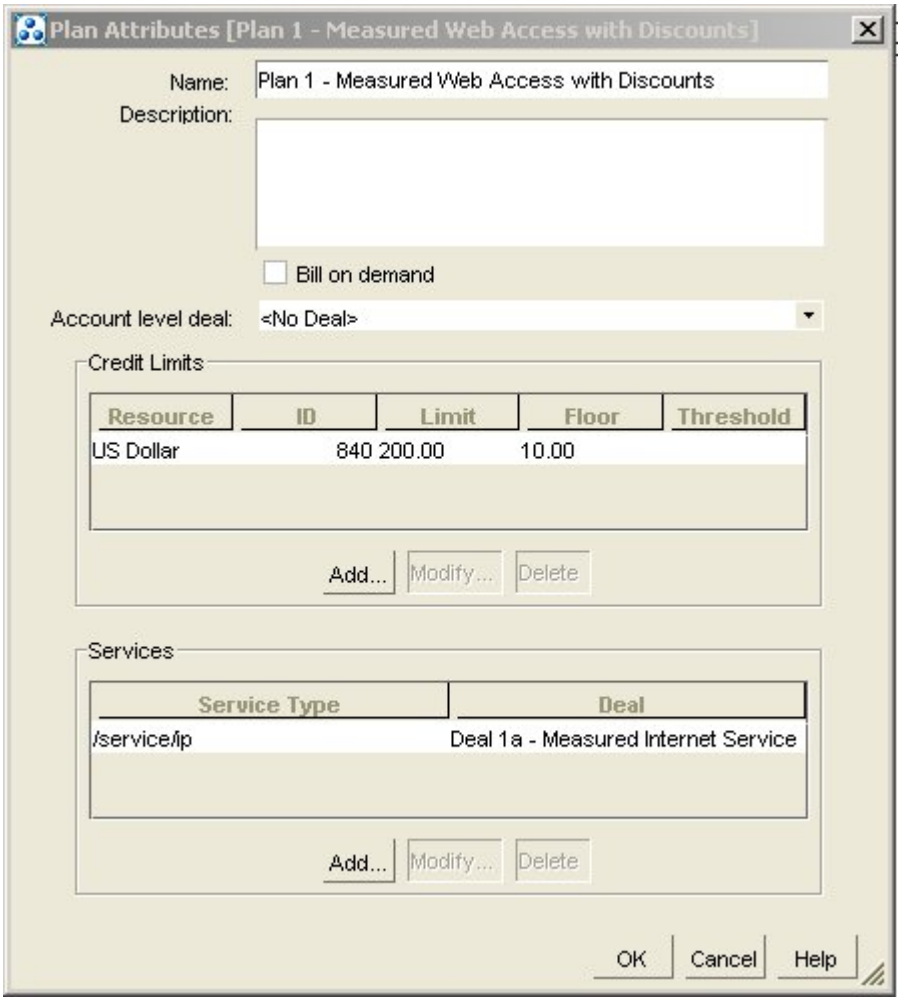
   To discount real-time rates, you can include a discount in the deal. Figure 11-15 shows a deal that provides a 50% discount for the first month cycle fee.

Figure 11-15 Discount Example



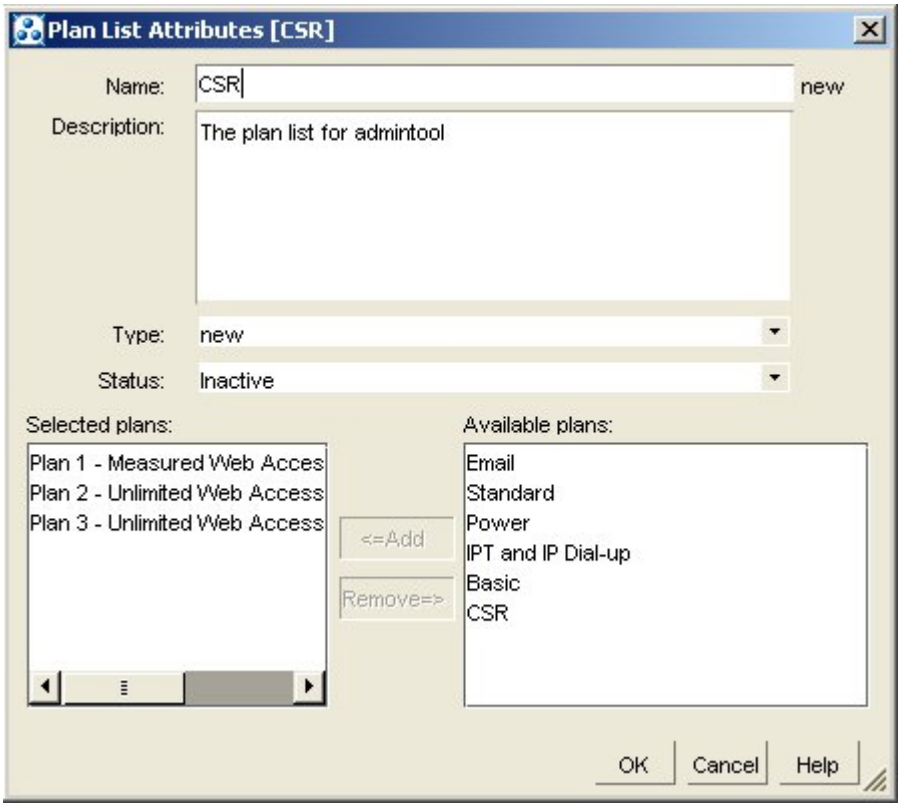
7. After defining deal-specific values for the products, you create a plan and add the deal to the plan. In the plan, you set the credit limit. In Figure 11-16, the credit limit is $200.

Figure 11-16 Plan Attributes Example



8. After creating the plan, you add it to one or more plan lists. In Figure 11-17, the plan is added to the **CSR - new** plan list. This plan list is displayed in Customer Center.

Figure 11-17 Plan List Attributes Example



9. After adding the plan to the plan list, choose **File - Commit** to commit the price list to the database.

   You test the price list by creating accounts that use your new plan, generating activity, and running billing to ensure that the account a balances are impacted correctly. See "Testing Your Price List".

Making Changes to Your Price List

You can make changes to your price list at any time. For example, if you offer a new service, you can create new plans and deals to charge for that service.

You can also change rates for existing products, add products to existing deals, and so forth.

Deleting Products

You cannot delete a product from the price list if it is owned by any account. To delete a product, cancel it in all accounts.

Logging Changes to Price Lists

To get notified when price lists change, BRM can create messages in the **cm.pinlog** file when price lists are updated. For information about **cm.pinlog** and other log files, see the discussion about using logs to monitor components in *BRM System Administrator's Guide*.

To log price list changes, set the **fm_rate log_refresh_product** entry in the Connection Manager (CM) **pin.conf** to **1**. If this entry is absent or set to **0**, BRM does not log changes to the price lists.

> **Note:**
>
> Products which are used while rating events are cached by rating. The cache is refreshed when a product is modified based on the CM **pin.conf** entry **refresh_product_interval**. This entry specifies the time after which the cache needs to be refreshed. If this entry is not specified, the default refresh interval is one hour. When the cache is refreshed and **log_refresh_product** is enabled, a debug message is created in the **cm.pinlog** file indicating the product POID that was refreshed.

To log price list changes:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Set the value for the **fm_rate log_refresh_product** entry to **1**.

```
- fm_rate log_refresh_product   1
```

3. Save the file.
4. Stop and restart the CM.

Ensuring Price List Consistency

For implementations covering a large geographic area, you might need regional price lists, each with variations in the pricing structure. In that case, you should maintain consistency in the names of price list elements.

For example, use a common name for noncurrency resources, such as access hours. It is easier to keep track of global usage when a resource has the same name on every regional price list. You should also have common names for usage rates.

Keep in mind that rates are applied independently of the products that include them. If multiple developers create products independently that can be purchased by the same customers, ensure that the rates in those products charge the same amounts and do not have conflicting priorities.

Troubleshooting a Price List

Pricing Center checks the validity of your price list. If you cannot save your price list, check for the following errors:

- **Time conflicts**. For example, you might set the start date for a rate to March 1 and then set the start date for a deal that contains the rate to February 1st. This means that on February 1st, a deal is available for purchase that contains a rate that is not valid for another month.
- **Nonvalid credit limits and credit floors**. A credit limit must be equal to or greater than 0. A credit floor must be less than or equal to the credit limit.

> **Note:**
>
> Pricing Center does not include or validate pipeline rating data when you save a price list.

Displaying Price List Elements

You can show a detailed outline of all the elements in a price list, including plans, deals, products, rate plans, and rates by running the **PriceList** report. For more information, see the discussion about price list report in *BRM Reports*.

> **Note:**
>
> Offer profiles cannot be displayed using Pricing Center.

Creating a Price List with the XML Pricing Interface

You can use the XML Pricing Interface instead of using Pricing Center to create your price list. See "Using the XML Pricing Interface to Create a Price List ".

Common Price List Solutions

The following topics describe how to implement these common price list solutions:

- Providing Free Hours
- Deleting Unused Free Hours
- Charging for Canceling a Product
- Charging a Discounted Purchase Fee Based on Date of Purchase
- Creating a “Third-Month Free" Rate
- Creating Discounts Based on Usage
- Creating Products for Administrative Events

> **Note:**
>
> These topics apply to real-time rating only.

Providing Free Hours

To provide 10 free hours every month, you need three rate plans:

- A cycle forward rate plan that credits 10 free hours every month.
- A usage rate plan with two rate tiers: one that charges hours as the resource until the free hours are gone and one that charges dollars as the resource.

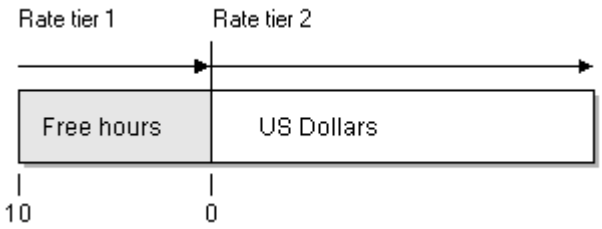
- A fold rate plan that deletes unused hours if the customer does not use all 10 free hours.

For example:

- Cycle forward rate plan: Credit 10 hours per month
- Balance impact: -10 hours every month
- Usage rate plan:
  - Rate tier 1: Charge hours. This tier is valid until the hours in the customer's account reaches 0.
  - Balance impact: 1 hour for every hour online
  - Rate tier 2: Charge dollars. This tier is valid when rate tier 1 can no longer be applied.
  - Balance impact: 1 dollar for every hour online.
- Fold rate plan: Delete unused hours (see "Deleting Unused Free Hours").

Figure 11-18 shows how the free hours are used first:

Figure 11-18 Free Hours Example



Deleting Unused Free Hours

When you credit free hours every month, you probably do not want your customers carrying over unused free hours from one month to the next.

To delete unused free hours, create a cycle fold rate plan that subtracts one hour for every free hour in the customer's balance at the end of the month. After the rate plan removes the left-over free hours, a cycle-forward rate plan applies the monthly credit of free hours.

Figure 11-19 shows the balance impact in Pricing Center. The resource is hours and the balance impact is -1, which removes one hour for every hour in the account balance.

Figure 11-19 Balance Impact that Deletes Unused Hours

	Resource ID	GLID	Impact Category	S	P	D	Fixed Amount	Scaled Amount	Units
1	async bulk hours [100...	undefine...		☐	☑	☐	0.00	-1.00	None [0]

Charging for Canceling a Product

To charge a fee for canceling a product, you include a cancel rate plan in your product.

To create a cancel rate plan that is applied only when closing an account, create a product that has only a cancel rate plan. You can cancel other products without charging a cancel fee, but

when the account is closed, the product with the cancel rate plan is canceled, which generates the cancel fee.

Charging a Discounted Purchase Fee Based on Date of Purchase

To charge a discounted purchase fee based on the date of purchase, you need two rate tiers:

- A high-priority rate tier that is valid only in a specified date range. This rate tier charges the discounted purchase fee.
- A low-priority rate tier that is always valid. This rate tier charges the normal purchase fee.

For example:

- Rate tier 1: Discounted purchase fee
- Priority: high
- Balance impact: 7.5 US dollars
- Absolute duration end time: 12:00 a.m. on 12/31/99
- Rate tier 2: Normal purchase fee
- Priority: low
- Balance impact: 15 US dollars

Creating a “Third-Month Free" Rate

To give a customer the third month of a cycle forward fee free, you must create a cycle forward rate plan with two rate tiers:

- Create a high-priority rate tier that uses relative duration to start three months after the product is purchased and has a balance impact of 0.
- Create a low-priority rate tier for the normal cycle forward fee.

For example:

- Rate tier 1: Free month
- Balance impact: 0 US dollars
- Relative duration start time: 55 days
- Relative duration end time: 65 days

  The relative duration start date is 55 days after purchase to ensure that the rate tier is valid before the third billing cycle starts, which, depending on the length of the month, could start within 59 days of the product purchase.

  The relative duration end date is 65 days after purchase to ensure that the rate tier is not valid in the fourth billing cycle.
- Rate tier 2: Normal cycle fee
- Balance impact: 19.95 US dollars

Creating Discounts Based on Usage

To define a rate plan based on volume usage, use multiple quantity discount brackets.

The following example uses two quantity discount brackets; one to charge .02 US dollars for each of the first 100 email messages and another .01 US dollars for each subsequent message.

Which quantity discount bracket is selected depends on the total quantity rated by both brackets.

- Quantity Discount Bracket 1: Balance impacts for first 100 email messages
- Minimum quantity: none
- Maximum quantity: 100
- Balance impact resource: US dollars
- Scaled impact: .02 US dollars
- Quantity Discount Bracket 2: Balance impacts for email messages after the first 100
- Minimum quantity: 100
- Maximum quantity: none
- Balance impact resource: US dollars
- Scaled impact: .01 US dollars

Creating Products for Administrative Events

To create products for administrative events, you must first define usage rates, as explained in "About Charging for Administrative Events".

When you create the products:

- If the event that you're rating is not tied to any service, use the All Accounts/No Services purchase level.
- If the event applies to all customers at all times, use the default product to define the rate.

Advanced Rating Features

For information about advanced rating features, see:

- Charging Cycle Forward Fees in Advance
- About Charging for Administrative Events
- About Using Product-Level Provisioning to Configure Services
- About Custom Event Analysis
- Setting Optional Rating Flags

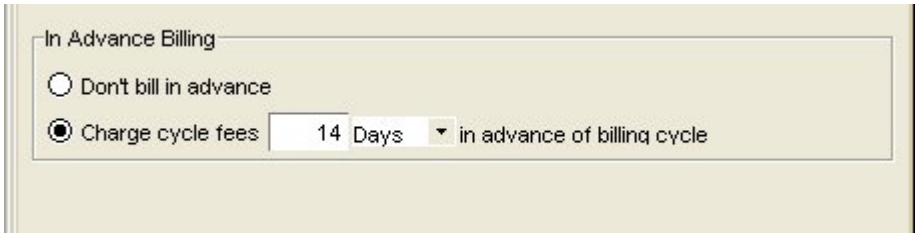
> **Note:**
>
> These topics apply to real-time rating only.

Charging Cycle Forward Fees in Advance

You can set up cycle forward rates to charge cycle forward fees in advance. This enables you to charge all or a portion of the next cycle fee in the current bill.

When you define your event map in Pricing Center, you can specify how far in advance to bill in days, weeks, or months, as shown in Figure 11-20:

Figure 11-20 Advance Billing Example



As part of setting up billing in advance, you must enable proration. Use Pricing Center to enable proration and take the other steps necessary to set up billing in advance.

Example: Charging Cycle Forward Fees in Advance for Monthly Cycle Fees

Suppose billing is monthly and in advance billing is set for 15 days.

When billing occurs on February 1, it charges cycle fee from February 1 to March 1 and additional 15 days to March 15.

When billing occurs on March 1, it charges cycle fee from March 15 to March 31 and additional 15 days to April 15.

When billing occur s on April 1, it charges cycle fee from April 15 to May 15, and so on.

Example: Charging Cycle Forward Fees in Advance for Quarterly Cycle Fees

Suppose billing is monthly and in advance billing is set for one month.

When billing occurs on February 1, it charges cycle fees from February 1 to May 1 and additional one month to June 1.

When billing occurs on March 1 and April 1, no cycle fees are charged.

When billing occurs on May 1, it charges cycle fees from June 1 to August 1 and additional one month to September 1.

When billing occurs on June 1 and July 1, no cycle fees are charged.

When billing occurs on August 1, it charges cycle fees from September 1 to December 1, and so on.

About Charging for Administrative Events

You can charge for *administrative events*, such as changing a password. To do so requires programming. See "About Charging for Custom Events and Attributes".

About Using Product-Level Provisioning to Configure Services

You can use product-level provisioning to define how a service is configured and to rate each configuration differently.

For example, if you have a product that provides an IP service, you can define two service configurations based on the bandwidth of the IP connection. You would create two products, one for each service configuration (for example, Fast Fax and Regular Fax).

To create products that configure services, you assign provisioning tags to the products. For example, to create a Fast Fax product, you would create a Fast_Fax provisioning tag that sets a higher bandwidth for the customer's connection. When a customer purchases the product that includes the Fast_Fax provisioning tag, the fax service is provisioned with the higher bandwidth.

When creating products in Pricing Center, you choose the provisioning tag that identifies the service configuration.

The advantage to configuring services by using products is that you do not need to define different services to handle different service configurations.

You can also use provisioning tags to create extended rating attributes, which enable you to offer special rates and promotions.

To create provisioning tags, see "Working with Provisioning Tags ".

About Remittance

Use the remittance feature to share the revenue you receive with third parties. You can direct BRM to calculate the amount of remittance in various ways. For example, you can pay a percentage of subscriber fees or a flat amount per new subscriber to third parties such as resellers or service providers.

You must create remittance products to use the remittance feature. See "Remitting Funds to Third Parties" in *BRM Configuring and Running Billing*.

About Custom Event Analysis

Custom event analysis enables you to perform rating based on:

- Custom attributes of events or non-event attributes, such as an account attribute or a profile attribute.
- Guidelines that are more complex than the defaults, such as rating based on percentage values or expressions containing a greater than or less than operator.

Setting up custom event analysis requires modifying or creating a policy opcode and editing and loading several configuration files.

Setting Optional Rating Flags

You can turn a number of optional rating features on and off by changing entries in the Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**):

- Changing the way fixed discount amounts are calculated. See "Providing Deal-Level Discounts".
- Turning credit floor checking on and off. See "Credit Limit and Floor Options".
- Determining whether to apply a credit limit when an account does not include a balance for a particular resource. See "Credit Limit and Floor Options".
- Determining whether folds for canceled (inactive) products should be rated.

- Determining whether the PIN_FLD_EXTENDED_INFO substruct field should be returned with the rating results. This field can be used to transport reusable information among transactions. For more information, see the input flist specification for PCM_OP_RATE_EVENT.

To use these features:

1. Open the *BRM_home*/**sys/cm/pin.conf** file.
2. Change the value of the **extra_rate_flags** entry.

   Each feature has a different hexadecimal value shown in Table 11-9.

Table 11-9 Optional Rating Features and Values

Optional Rating Feature	Value	Effect
Fixed discount option	0x01	If present, discounts are calculated using this formula: **[(rate – discount) * quantity]** If absent, discounts are calculated using this formula: **[(rate* quantity) – discount]**
Credit floor checking	0x02	If present, credit floor checking is disabled. If absent, credit floor check is enabled.
Return extend information	0x10	If present, extended information is returned with the rating result. If absent, extended information is not returned.

To use more than one option, add the values for each option and use the sum value.

You do not need to restart the CM to enable this entry.

12

Understanding the Sample Pricing Plans

This chapter describes the sample pricing plans included with Oracle Communications Billing and Revenue Management (BRM) Pricing Center.

Topics in this document:

For information about creating a price list, see "About Creating a Price List".

About the Sample Plans

The following sections describe the sample price plans.

Purpose of the Sample Plans

The sample BRM pricing plans described in this chapter show a range of possibilities for creating typical plans with BRM. Also included are some advanced sample plans to illustrate pricing and rating strategies. You can use the samples as templates for creating your own pricing plans. You might find a plan that matches exactly the plan you want to offer to your customers. More likely, you will see elements of several plans that you want to include in your own plan. You can pick and choose which elements you want and then see how those elements were created, making it easy to incorporate them into your plan.

Looking at the Sample Plans

BRM provides price list documents (files with the IPL or XML extension) that you can display in the Pricing Center:

> **Note:**
>
> Offer profile data cannot accessed using Pricing Center. Use the **loadpricelist** utility or the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE opcode.

- Basic BRM plans for Internet service providers (**basicISP.ipl**) are located by default in the **Sample_Price_Plans** subdirectory of the Pricing Center installation directory.
- Plans that are based on a BRM optional manager are located by default in the **Optional_Manager_Plans** subdirectory of the Pricing Center installation directory.

> **Note:**
>
> If you're implementing an optional manager, it is a good idea to familiarize yourself with the basic plans before creating a price list for an optional manager.

- The advanced sample plans are located by default in the **Sample_Price_Plans** subdirectory of the Pricing Center installation directory.

Opening an IPL Price List File

1. Start Pricing Center.
2. Choose **File - Open**.
3. Open the **PricingCenter\Sample_Price_Plans** directory.
4. Select the IPL file and click **Open**.

Opening an XML Price List File

1. Start Pricing Center.
2. Choose **File - Import > Real-time Data**.
3. Select the XML file and click **Open**.

   You must connect to a database to access configuration information, such as resources, G/L IDs, and valid events for services.
4. Enter the database information in the Welcome dialog box.

> **Note:**
>
> Click the **Help** button for information on login values.

Descriptions of the Sample Plans

This section describes each of the sample plans in the language your company's marketing department might present to your customers. Each plan includes a brief description, the target customer, and the key elements of the plan.

For step-by-step procedures on creating these plans, see "Re-Creating the Sample Plans".

Plans for Internet Service Providers

This section describes the sample plans for internet service providers.

Plan 1 – Measured Web Access with Discounts

This plan provides Internet access for a basic monthly fee, $9.95 in this example, plus an hourly fee that is discounted for usage outside of prime time.

Target customer

New Web user who might adjust access times to take advantage of discount periods

Concepts illustrated

- Simple monthly (cycle forward) fee for Internet access
- Prorating of cycle fees
- Measured rates
- Time-of-day usage rates
- Priority assignments for multiple rates
- Product purchase limited to a set maximum

Key elements of plan

- Internet access service
- $9.95 basic monthly rate
- Measured rate of $2.00/hour for prime-time access (M-F, 8 a.m. to 6 p.m.)
- Measured rate of $1.00/hour for access outside of prime time
- First month prorated
- No partial refund for cancellation in mid-cycle
- Email service
- A single email address available for an additional $3.00/month

To re-create this plan

See "Re-Creating Plan 1 – Measured Web Access with Discounts".

Plan 2 – Unlimited Web Access with Discounts

This plan provides unlimited Internet access and one email address for a fixed monthly fee, $19.95 in this example. There's an initial sign-up fee, discounted when the plan is purchased before a certain date. The third month is free.

Target customer

Heavy-duty Internet user who might be encouraged to stay with the plan by the offer of discounts

Concepts illustrated

- Monthly (cycle forward) fees for Internet access
- Purchase fee
- Discount period on purchase
- Single free period

Key elements of plan

- Internet access service

- Sign-up fee of $15.00
- Sign-up fee discounted by 50% through May, 2000
- $19.95 monthly rate, billed on the first day of the month
- Third month free
- One email address included

To re-create this plan

See "Re-Creating Plan 2 – Unlimited Web Access with Discounts".

Plan 3 – Unlimited Internet Service with Recurring Discounts

This plan provides unlimited Internet access and one email address for a fixed quarterly fee of $45, payable at the beginning of each quarterly period. For every year with this plan, there's a bonus of one free month.

Target customer

Frequent Internet user who might pay in advance for a lower basic rate and who might be encouraged to stay with the plan by the offer of recurring discounts

Concepts illustrated

- Multi-month (cycle forward) fee
- Measured fees
- Recurring free period
- Custom BRM resources
- Reusing products from other plans

Key elements of plan

- Internet access service
- $45.00 per quarter, billed on the first day of every third month
- Last month of each year is free
- One email address included

To re-create this plan

See "Re-Creating Plan 3 – Unlimited Internet Service with Recurring Discounts".

Re-Creating the Sample Plans

This section shows how each of the sample pricing plans was created by using Pricing Center. For more information on specific procedures, see Pricing Center Help.

Overview of Creating a Pricing Plan

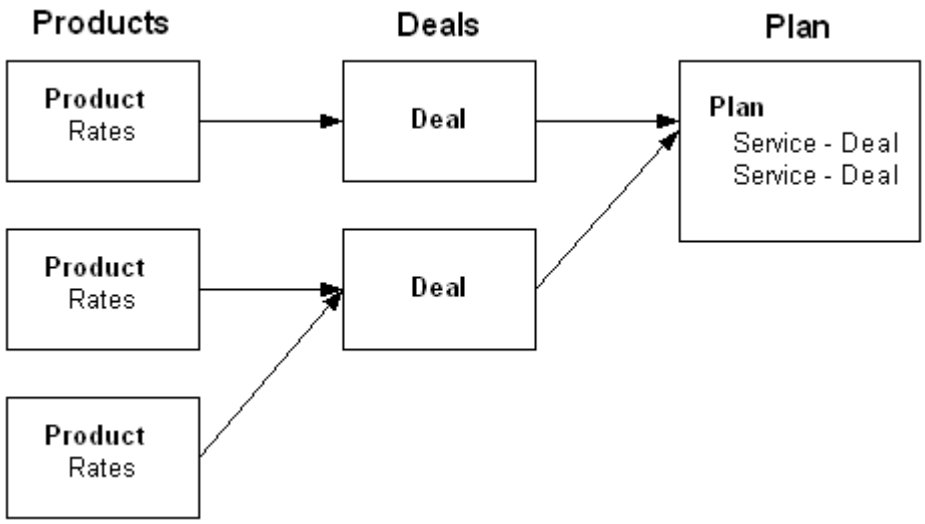
Your company's marketing department writes a proposal for offering services to your customers. You create a plan that translates that proposal into the language that BRM uses to keep track of the rates and then bill for customer usage.

The *pricing plan* shows one or more services that you offer your customers and the terms under which you provide those services. The terms and conditions of a pricing plan is called a *deal*. Each deal consists of one or more *products*, which show the rates charged for a service. You use products to map rates to events, so that when the event occurs in the BRM system, the rate is charged.

To create a plan, you define the products and then add them to a deal, which you bundle into a plan and associate with your service. In some cases you might also need to create resources or impact categories for a plan.

Figure 12-1 shows the building blocks of a pricing plan:

Figure 12-1 Pricing Plan Building Blocks



To create a plan, follow these general steps:

1. Define the key elements of the plan: what services you want to offer and what you want to charge for them.

   The descriptions of the sample plans show the key elements of these plans.

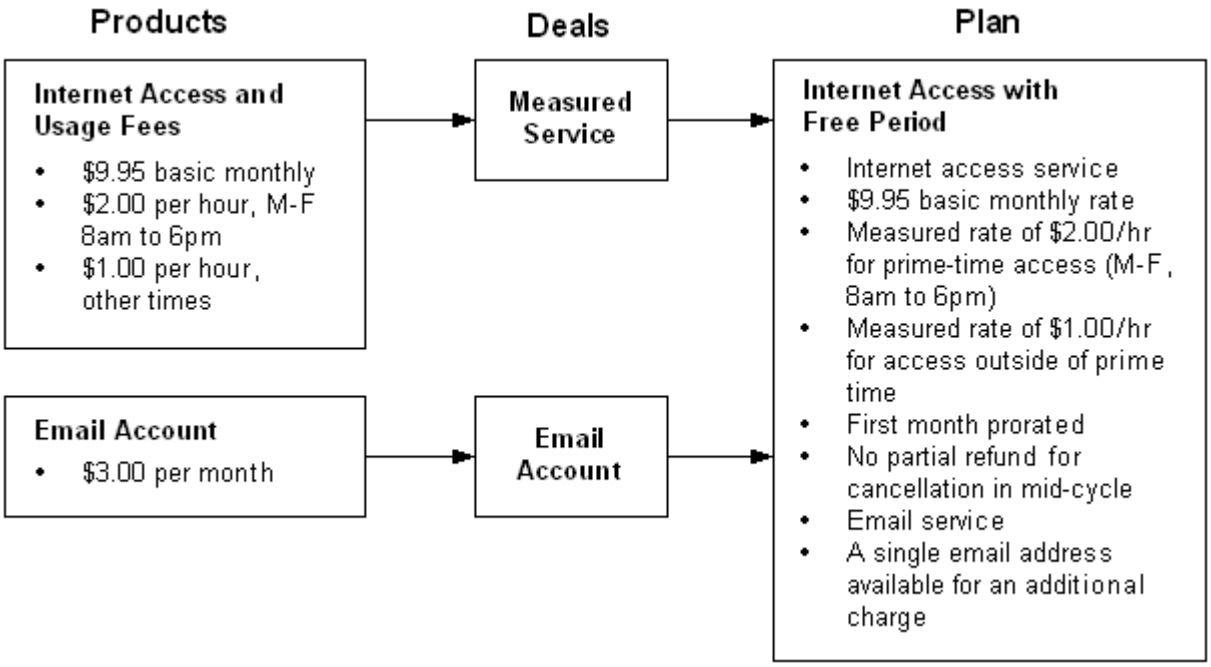
2. Use Pricing Center to create a product for each key element of your plan. Usually, each service you offer uses one product to contain all rates for that service.
3. Use Pricing Center to group the products into one or more deals.
4. Use Pricing Center to create a pricing plan that associates the deals with the services.

Plans for Internet Service Providers

Figure 12-2 show the Measured Web Access with Discounts Plan used in the following example.

Re-Creating Plan 1 – Measured Web Access with Discounts

Figure 12-2 Measured Web Access with Discounts Plan



Putting together Plan 1

1. Create a product to charge for the Internet access service and Internet measured usage. See "Product 1a – Internet Access".
2. Create a product to charge for the email account. See "Product 1b – Email Account".
3. Create a deal for the Internet access. See "Deal 1a – Measured Internet Service".
4. Create a deal for the email account. See "Deal 1b – Standard Email Access".
5. In Pricing Center, choose **Insert – Plan**.
6. Name the plan: **Plan 1 – Measured Web Access with Discounts**. (You can also add a description.)
7. In the **Services** section, click **Add**.
8. For **Service Type**, select **/service/ip**.
9. For **Deal**, select **Deal 1a – Measured Internet Service**.
10. Click **OK**.
11. In the **Services** section, click **Add**.
12. For **Service Type**, select **/service/email**.
13. For **Deal**, select **Deal 1b – Standard Email Access**.
14. Click **OK**.
15. In the **Credit Limits** section, click **Add**.

16. In the **Resource ID** list, select **US Dollar**.
17. In the **Credit Limit** box, type **250**.
18. Double-click **OK** to finish the plan.

Product 1a – Internet Access

The purpose of this product is to charge for the Internet access service at the beginning of each month and for time spent connected to the Internet when it occurs. The product has three rates:

- One for the monthly fee
- One for usage during prime-time hours ($2.00/hour)
- One for usage during off-peak hours ($1.00/hour)

You create the fees by mapping the rates to *events*, which define how the charges occur.

Product 1a - Defining the Product

Table 12-1 describes the steps for defining the product.

Table 12-1 Defining the Product

Do This	Why
In Pricing Center, choose **Insert – Product** and select **Create a new product**.	To create a product.
Click **Next** and give the product a name: **Product 1a – Internet Access**. (You can also add a description.) Click **Next**.	The product name identifies the product in Pricing Center. This name must be unique to all products in the same database brand.
When asked what type of product to create, select **Subscription** and in **Applies To**, choose **/service/ip**.	This enables the product to contain recurring monthly and usage rates and associates it with the IP service.
Finish the product creation wizard, and click **Yes** at the end. The Product Attributes window opens.	This product uses the default values so you do not need to specifically set them.

Product 1a - Creating the Monthly Internet Fee

You must create an event rating map to charge the customer an access fee each month for the IP service. The access fee is mapped to the monthly cycle event so fees can be tracked and posted to the customer's account when the cycle event occurs.

Table 12-2 describes the steps for creating the Monthly Internet Fee.

Table 12-2 Creating the Monthly Internet Fee

Do This	Why
In the **Event Map** area of the Product Attributes window, click **Add**. In the **Event** column, select **Monthly Cycle Forward Event**.	To create the monthly access fee by associating the rate details with the cycle event, as specified by the plan. When you select **Monthly Cycle Forward Event**, Pricing Center automatically selects **Occurrence** in the **Measured By** column and **Single Rate Plan** in the **Rate Plan Structure** column. Leave these values as they are.
Click **Open Rate Plan**.	To set up the rates for this product.
In the **Plan Detail** tab, enter **cycle_forward_monthly** in the **Name** field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose.
In the **Rate Structure** tree view, select **Tier 1**. In the **Tier Detail** tab, enter a tier name: **Monthly Service Fee**.	The name identifies the rate tier in the product and reminds you when it is valid.
In the **Tier Detail** tab, select either **Absolute** or **Relative**, then select **Starts Immediately** and **Never Ends**. **Note:** When you select **Starts Immediately** and **Never Ends**, it does not matter whether you select **Absolute** or **Relative** effective dates.	To make the rate available indefinitely, so it never expires in the BRM database.
In the **Rate Structure** tree view, select **Rate 1**. In the **Rate Detail** tab, enter a rate name: **$9.95 a month**. **Note:** Rate names must be unique. In the **/rate** storable class, the rate name is specified in the PIN_FLD_DESCR field.	To remind you of the purpose of this rate.
In the **Rate Data** tab, select **Overrides credit limit**.	To continue to charge for Internet access when customers exceed the credit limit.
In the **Rate Data** tab, select **Continuous** in the **Based on** list.	This option is the default when you use only one group of balance impacts to rate an event.
Click the **Proration** tab.	To specify how to prorate the charges.
In the **Purchase Proration Information** group, click **Calculate the charge based on the amount used**.	To charge customers only for the portion of the first month when they owned the product. **Note:** Most companies base the monthly accounting cycle on the day the customer purchases the product rather than on a calendar month, so proration is not necessary.
In the **Cancel Proration Information** group, click **Charge for the entire cycle**.	To specify that the customer is charged the full monthly fee, even if the customer cancels the service in the middle of the accounting cycle.
In the **Rate Structure** tree view, select the item at the bottom of the rate plan tree to show the **Balance Impacts** tab.	To define the cost of the cycle rate.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the impacts list and to specify the balance impacts - what you charge the customer for the service.

Table 12-2 (Cont.) Creating the Monthly Internet Fee

Do This	Why
Click **Resource ID** and scroll to select **US Dollar [840]**.	To bill in US dollars.
Click the **GLID** and scroll to select **Monthly Fees [102]**.	To record the charges as monthly fees in your general ledger.
Click the **S** check box to select **Sponsorable**.	To enable the rate to be paid by another account. Select this option to enable one account to pay for another.
Click the **P** check box to select **Proratable**.	Although you already specified the terms of proration, you now confirm that this rate can be prorated.
Click the **D** check box to select **Discountable**.	To enable the rate to be discounted when the service is purchased.
In the **Scaled Amount** column, type **9.95**.	Recurring charges, such as the monthly fee in this product, are called *scaled* rates. Enter the dollar amount of the charge, which is $9.95 per month.
In **Units**, select **None [0]**.	This rate uses the **Occurrence** ratable usage metric (RUM), which is not measured in units.
Click **OK**.	You are finished creating the monthly fee and now must create the usage fees.

Product 1a - Creating the Measured Usage Fees

You must create an event rating map to charge the customer for time spent connected to the Internet. You map the hourly rates to an IP session event so fees can be tracked and posted to the customer's account when the customer uses the service. Although the rates are mapped to the same event, you differentiate between the two by specifying the available time for the rates: one for usage during prime-time hours ($2.00/hour) and one for usage during off-peak hours ($1.00/hour).

Table 12-3 describes the steps for creating the Measured Usage Fees.

Table 12-3 Creating the Measured Usage Fees

Do This	Why
In the **Event** group, click **Add**. In the **Event** column, select **Session Event**.	To associate rate details with the IP session event, so fees can be tracked and posted to the correct account when customers connect to the Internet, as specified by the plan. When you select **Session Event**, by default Pricing Center selects **Duration** in the **Measured By** column and **Single Rate Plan** in the **Rate Plan Structure** column. Do not change these values.
Click **Advanced**.	To display the Event Attributes dialog box.
In **Time of day mode**, select **Timed**.	To specify that the fees depend on the length of the session.
In **Time zone mode**, select **Server**.	To specify that this event will be rated by using the time zone in which the BRM server is located.

Table 12-3 (Cont.) Creating the Measured Usage Fees

Do This	Why
In **Minimum event quantity**, enter **1** and select **Minutes** as the unit.	To set the minimum session time, even if the session lasts less than a minute.
In **Rounding Increment**, enter **1** in **Units** and select **Minutes** as the unit. In **Rounding rule**, select **Up**.	To round the session length up to the next whole minute, so seconds are not recorded. For example, if a session lasts 1 minute and 10 seconds, the session is recorded as 2 minutes.
Click **OK**.	To return to the **Product Attributes** dialog box.
In the **Single Rate Plan** group, click **Open Rate Plan**.	To set up the rates for this product.
In the **Plan Detail** tab, enter **Internet Usage Fee** in the **Name** field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose.
In the **Rate Structure** tree view, select **Tier 1**. In the **Tier Detail** tab, enter a name for the rate tier: **Hourly Fee for Basic Internet Service.**	The name identifies the rate tier in the product and reminds you when it is valid.
In the **Tier Detail** tab under **Effective Dates**, select **Absolute**.	To enable the time-of-day settings for the rate.
In the **Use Time/Day Restrictions** group, select **Time restrictions**. (In the **Valid Days** tab, leave the default settings.)	To add a time tier to the rate.
In the **Rate Structure** tree view, select **New Time of Day Range**. At the top of the **Valid Times** tab, replace the **New Time of Day Range** text with a name for the hourly range: **8am-6pm**.	The name identifies the level in the rate tier and reminds you when it is valid.
Leave the start time as **08:00 AM**, click the **08:00 PM** end time control and use the down arrow to set **06:00 PM (18:00 hours)**.	This sets the rate as valid from 8 a.m. to 6 p.m.
Under **Rate Structure,** select **Rate 1. In the Rate Data tab, enter** a name for the rate: **$2 per hour**.	The name identifies the amount of the fee in the 8 a.m. to 6 p.m. rate tier.
Select **Overrides credit limit**.	To continue to charge for Internet access if customers exceed their credit limit.
In the **Based on** list, select **Continuous**.	This option is used to rate time-of-day events because the amount charged by the first rate tier can affect the which balance impacts are used in the second rate tier.
In the **Rate Structure tree view,** select **No Minimum - No Maximum** to display the **Balance Impacts** tab.	This displays the **Balance Impacts** tab, where you define the cost of the rate.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Under **Balance Impacts**, click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the list of balance impacts and to specify the balance impacts **-** what you charge the customer for the service.
In the **Resource ID** list, select **US Dollar [840]**.	To bill in US dollars.

Table 12-3 (Cont.) Creating the Measured Usage Fees

Do This	Why
In the **GLID** list, select **Dialup Usage Fee [104]**.	To record the charges as Internet usage fees in your general ledger.
Click **S for Sponsorable**.	To enable the rate to be paid by another account. Select this option to enable one account to pay for another.
In the **Scaled** box, type **2**.	Hourly fees are considered *scaled* rates. You enter the dollar amount of the charge per hour.
In **Units**, select **Hour [3]**.	This is the increment you use to measure and charge for the usage. You are finished creating this rate, which applies between 8:00 a.m. and 6:00 p.m.
In the **Rate Structure** tree view, select the **Hourly Fee for Basic Internet Service** tier and click **Add**.	To create the second (off-peak) usage fees.
At the top of the **Valid Times** tab, replace the **New Time of Day Range** text with a name for the hourly range: **6pm-8am**. Press **Enter**.	The name identifies the valid time of the rate.
Click the **08:00 AM** start time control and set it to **06:00 PM (18:00 hours)**. Then set the end time control to **08:00 AM**.	This sets the rate as valid from 6 p.m. to 8 a.m.
In the **Rate Structure** tree view under **6pm - 8am**, select **New Rate**.	This displays the Rate Data tab, where you define the fees for off-peak hours.
In the Rate Data tab, enter a rate name: **$1 per hour**.	The name identifies amount of the 6 p.m. - 8 a.m. rate tier.
Select **Overrides credit limit**.	To continue to bill for Internet access if customers exceed the credit limit.
In the **Based on** list, select **Continuous**.	This option is used to rate time of day events because the amount charged by the first rate tier can affect the discount bracket used in the second rate tier.
In the **Rate Structure** tree view, under the rate name **$1 per hour**, select **No Maximum - No Minimum**.	To display the **Balance Impacts** tab.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	There is no minimum or maximum beyond which this rate would not be valid.
Under **Balance Impacts**, click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the list of impacts and to specify the balance impacts **-** what you charge the customer for the service.
In the **Resource ID** list, select **US Dollar [840]**.	To charge in US dollars.
In the **GLID** list, select **Dialup Usage Fee [104]**.	To record the fees as dialup usage fees in your general ledger.
Click **S for Sponsorable**.	To enable the rate to be paid by another account. Select this option to enable one account to pay for another.

Table 12-3 (Cont.) Creating the Measured Usage Fees

Do This	Why
In the **Scaled Amounts** field, type **1**.	Hourly charges are considered *scaled* rates. You enter the dollar amount of the charge per hour. You are finished specifying the rate for the weekday range. Now you must create the off-peak rate that is valid on the weekends.
Click **OK** until you return to the Document window.	You are finished specifying the usage fees for this product.

Product 1b – Email Account

The purpose of this product is to charge for the single email account available with this plan.

Table 12-4 describes the steps for defining the product.

Product 1b - Defining the Product

Table 12-4 Defining the Product

Do This	Why
In Pricing Center, choose **Insert – Product** and select **Create a new product**.	To create a product.
Click **Next** and give the product a name: **Product 1b – Email Account**. (You can also add a description.) Click **Next**.	The product name identifies the product in Pricing Center. This name must be unique to all products in the same database brand.
When asked what type of product to create, select **Subscription** and in **Applies To**, choose **/service/email**.	This enables the product to contain recurring monthly rates and associates it with the email service.
Continue through the product creation wizard until you get to the **Valid Purchase Quantity** group.	Although you can change various attributes of the product while using the wizard, this sample product uses the default attributes up to this point.
Click **Maximum of** and enter **1**.	To specify that a customer can purchase no more than one email account.
Finish the product creation wizard, and click **Yes** at the end.	After you finish the wizard, you specify the rates for this product.

Product 1b - Creating the Monthly Email Fee

You must create an event rating map to charge the customer a fee each month for the email service. The email fee is mapped to the monthly cycle event so fees can be tracked and posted to the customer's account when the cycle event occurs.

Table 12-5 describes the steps for creating the Monthly Email Fee.

Table 12-5 Creating the Monthly Email Fee

Do This	Why
In the **Event Map** group, click **Add**. In the **Event Type** column, select **Monthly Cycle Forward Event**.	To create the monthly access fee by associating the rate details with the cycle event, as specified by the plan. When you select **Monthly Cycle Forward Event**, by default Pricing Center selects **Occurrence** in the **Measured By** column. You cannot change this value. **Note:** The values in the **Rating Details** group are only used for time-dependent fees.
In the **Rate Plan Structure** column, select **Single Rate Plan**. Under **Single Rate Plan**, click **Open Rate Plan**.	To set up the rates for this product.
In the **Plan Detail** tab, enter **cycle_forward_monthly** in the **Name** field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose.
In the **Rate Structure** tree view, select **Tier 1**. In the **Tier Detail** tab, enter a name for the rate tier: **Monthly Email Fee**.	The name identifies the rate tier in the product and reminds you when its valid.
Under **Tier Detail**, select **Absolute** or **Relative**, and then **Starts Immediately** and **Never Ends**. Also, under **Use Time/Day Restrictions**, select **No restrictions**.	To make the rate available indefinitely, so it never expires in the BRM database and the fee is always charged.
In the **Rate Structure tree view,** select **Rate 1. In the Rate Detail tab, enter** a name for the rate: **$3 a month**. **Note:** Rate names must be unique. In the **/rate** storable class, the rate name is specified in the PIN_FLD_DESCR field.	To remind you of the purpose of this rate.
Select **Overrides credit limit**.	To continue to charge for the email account when customers exceed the credit limit.
In the **Based on** list, select **Continuous**.	This option is the default when you use only one quantity discount bracket to rate an event.
Click the **Proration** tab.	To specify how to prorate the charges.
In the **Purchase Proration Information** group, click **Calculate the charge based on the amount used**.	To charge customers only for the portion of the first month when they owned the product. **Note:** Most companies base the monthly accounting cycle on the day the customer purchases the product rather than on a calendar month, so proration is not necessary.
In the **Cancel Proration Information** group, click **Charge for the entire cycle**.	To specify that the customer is charged the full monthly fee, even if the customer cancels the service in the middle of the accounting cycle.
In the **Rate Structure tree view under $3 a month**, select **No Minimum - No Maximum** to display the **Balance Impacts** tab.	This displays the **Balance Impacts** tab, where you define the fee.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Under **Balance Impacts**, click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the impacts list and to specify the balance impacts **-** what you charge the customer for the service.

Table 12-5 (Cont.) Creating the Monthly Email Fee

Do This	Why
Click the **Resource ID** box and scroll to select **US Dollar [840]**.	To charge in US dollars.
In the **GLID** list, select **Monthly Fees [102]**.	To record the charges as monthly fees in your general ledger.
Click **S for Sponsorable**.	To enable the rate to be paid by another account. Select this option to enable one account to pay for another.
In the **Scaled** box, type **3**.	Recurring fees, such as the monthly fee in this product, are called *scaled* rates. You enter the dollar amount of the fee.
In **Units**, select **None [0]**.	This rate uses the **Occurrence** ratable usage metric (RUM), which is not measured in units.
Click **OK** until you return to the Document window.	You are finished creating the email product.

Deal 1a – Measured Internet Service

1. In Pricing Center, choose **Insert – Deal**.
2. Name the deal: **Deal 1a – Measured Internet Service**. (You can also add a description.)
3. In **Applies to**, select **/service/ip**.
4. In the **Products Associated with Deal** group, click **Add**.
5. In the **Product Name** list, select **Product 1a – Internet Access**.
6. Click **OK**.

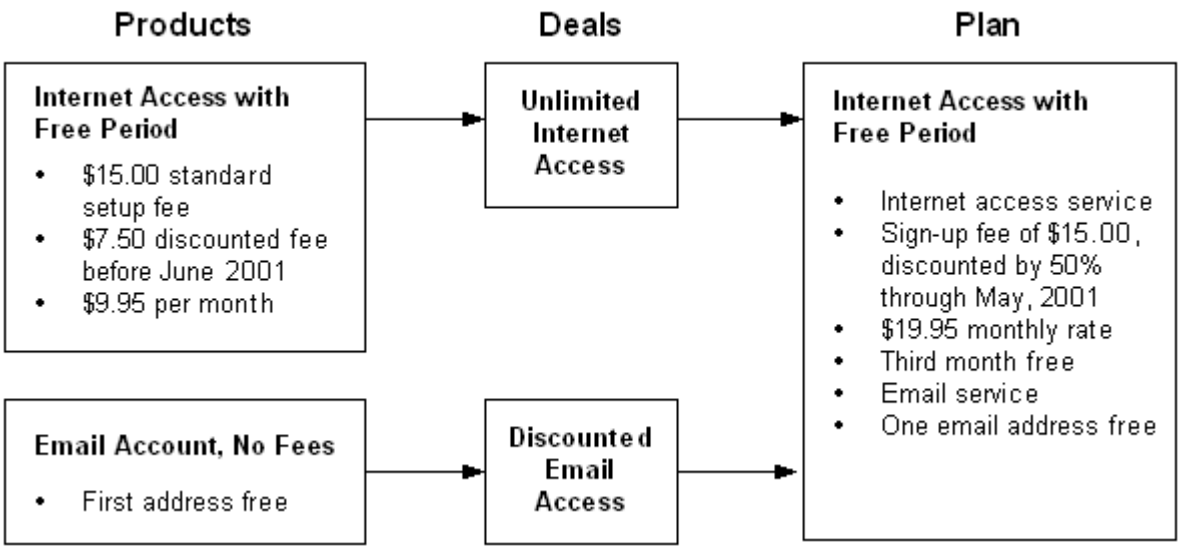
Deal 1b – Standard Email Access

1. In Pricing Center, choose **Insert – Deal**.
2. Name the deal: **Deal 1b – Standard Email Access**. (You can also add a description.)
3. In **Applies to**, select **/service/email**.
4. In the **Products Associated with Deal** group, click **Add**.
5. In the **Product Name** list, select **Product 1b – Email Account**.
6. Click **OK**.

Re-Creating Plan 2 – Unlimited Web Access with Discounts

Figure 12-3 show the Unlimited Web Access with Discounts Plan used in the following example.

Figure 12-3 Unlimited Web Access with Discounts Plan



Putting together Plan 2

1. Create a product to charge the sign-up fee and monthly Internet access fee. See "Product 2a – Internet Access with Free Period".
2. Create a product for the email account. See "Product 2b – Email Account, No Fees".
3. Create a deal for the Internet access. See "Deal 2a – Unlimited Internet Service".
4. Create a deal for the email account. See "Deal 2b – Discounted Email Access".
5. In Pricing Center, choose **Insert – Plan**.
6. Name the plan: **Plan 2 – Unlimited Web Access with Discounts**. (You can also add a description.)
7. In the **Services** section, click **Add**.
8. For **Service Type**, select **/service/ip**.
9. For Deal, select **Deal 2a – Unlimited Internet Service**.
10. Click **OK**.
11. In the **Services** section, click **Add**.
12. For **Service Type**, select **/service/email**.
13. For Deal, select **Deal 2b – Discounted Email Access**.
14. Click **OK** to finish the plan.

Product 2a – Internet Access with Free Period

The purpose of this product is to charge the customer at the time of registration, and then monthly for Internet access.

Product 2a - Defining the Product

Table 12-6 describes the steps for defining the product.

Table 12-6 Defining the Product

Do This	Why
In Pricing Center, choose **Insert – Product** and select **Create a new product**.	To create a product.
Click **Next** and give the product a name: **Product 2a – Internet Access with Free Period**. (You can also add a description.)	The product name identifies the product in Pricing Center. This name must be unique to all products in the same database brand.
When asked what type of product to create, select **Subscription** and in **Applies To**, choose **/service/ip**.	This enables the product to contain recurring monthly rates and associates it with the IP service.
Continue through the rest of the product creation wizard, and click **Yes** at the end.	Although you can change other attributes of the product while using the wizard, this sample product uses the remaining default attributes.

Product 2a - Creating the Sign-up Fees

You must create an event rating map so the sign-up fees are charged when the purchase event occurs. You use two rate tiers to define the fees. The first rate tier is for a discounted sign-up fee of $7.50, and the second rate tier is for a standard sign-up fee of $15.

Product 2a - Creating the Discounted Sign-up Fee

You want BRM to use the $7.50 discounted rate until it expires. Creating it first in the rate plan gives it a higher priority and ensures that BRM uses this rate during the valid discount period, instead of defaulting to the standard sign-up fee.

Table 12-7 describes the steps for creating the discount sign-ip fee.

Table 12-7 Defining the Product

Do This	Why
In the **Event Map** group, click **Add**. In the **Event** column, select **Product Purchase Fee Event**.	To create the sign-up fee by associating the rate details with the purchase event, as specified by the plan. When you select **Product Purchase Fee Event**, Pricing Center automatically selects **Occurrence** in the **Measured By** column. You cannot change this value. **Note:** The values in the **Rating Details** group are used only for time-dependent fees.
In the **Rate Plan Structure** column, select **Single Rate Plan**. Under **Single Rate Plan**, and click **Open Rate Plan**.	To set up the rates for this product.
In the **Plan Detail** tab, enter **purchase** in the **Name** field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose. For system events, you must enter the exact name of the rate plan.
In the **Rate Structure** tree view, select **Tier 1**. In the **Tier Detail** tab, enter a name for the rate tier: **Discounted IP Sign-up Fee**.	The name identifies the rate tier in the product and reminds you that it is a discounted fee.

Table 12-7 (Cont.) Defining the Product

Do This	Why
Under Effective Dates, select **Absolute**.	To make the rate available for a limited time, and to ensure that it expires on a certain date.
Under **Ends**, select **At**, then **On**, and set the end date to **03/01/2001**.	The discount is valid through the end of February, 2001. Valid date ranges do not include the end date; therefore, you enter the first date when this rate is no longer valid.
Under **Use Time/Day Restrictions**, select **No restrictions**.	To bypass setting valid days of the week.
In the **Rate Structure** tree view, select **Rate 1**. In the **Rate Data** tab, enter a rate name: **$7.50 before 3/1/01**. **Note:** Rate names must be unique. In the **/rate** storable class, the rate name is specified in the PIN_FLD_DESCR field.	To remind you of the purpose of this rate.
Select **Overrides credit limit**.	To continue to charge for Internet access when customers exceed the credit limit.
In the **Based on** list, select **Continuous**.	This option is the default when you use only one quantity discount bracket to rate an event.
In the **Rate Structure** tree view, select **No Minimum -No Maximum**.	This displays the **Balance Impacts** tab, where you define the cost of the rate.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Under **Balance Impacts**, click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the impacts list and to specify the balance impacts **-** what you charge the customer for the service.
Click the **Resource ID** row and scroll to select **US Dollar [840]**.	To charge in US dollars.
Click the **GLID** row and scroll to select **Purchase Fees [101]**.	To record the fees as purchase fees in your general ledger.
Select **S for Sponsorable**.	To enable the fee to be paid by another account. Select this option to enable one account to pay for another.
In the **Fixed Amount** field, type **7.5**.	One-time charges, such as the purchase fee in this product, are called *fixed* rates. You enter the dollar amount of the charge.
In the **Rate Structure** tree view, select **Purchase**.	You are finished specifying this rate tier.

Product 2a - Creating the Standard Sign-up Fee

You create the standard sign-up fee by using a second rate tier in the purchase rate plan. Creating the standard rate after creating the discounted rate gives the standard rate a lower priority, which ensures that the discount rate is used during the valid discount period.

Table 12-8 describes the steps for creating the standard sign-up fee.

Table 12-8 Creating the Standard Sign-up Fee

Do This	Why
In the **Rate Structure** tree view, select **Purchase** and then click **Add**.	To add a rate tier to the rate plan.
In the **Rate Structure** tree view, select **New Rate Tier**. In the **Tier Detail** tab, enter a name for the rate tier: Standard.	The name identifies the rate tier in the product and reminds you of the purpose of this rate.
Under **Tier Detail**, select **Absolute** or **Relative**, then **Starts Immediately** and **Never Ends**. Also, under **Use Time/Day Restrictions**, select No restrictions.	To make the rate available indefinitely, so it never expires in the BRM database.
In the **Rate Structure** tree view, select **New Rate**. In the **Rate Data** tab, enter a name for the rate: **$15 Sign-up Fee**. **Note:** Rate names must be unique. In the **/rate** storable class, the rate name is specified in the PIN_FLD_DESCR field.	To remind you of the purpose of this rate.
Select **Overrides credit limit**.	To continue to charge for the email account if customers exceed the credit limit.
In the **Based on** list, select **Continuous**.	This option is the default when you use only one quantity discount bracket to rate an event.
In the **Rate Structure** tree view, select **No Minimum - No Maximum**.	This displays the **Balance Impacts** tab, where you define the cost of the purchase rate.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Under **Balance Impacts**, click **Add**.	To add a row to the impacts list and to specify the balance impacts **-** what you charge the customer for the service.
Click the **Resource ID** box and scroll to select **US Dollar [840]**.	To charge in US dollars.
In the **GLID** list, select **Purchase Fees [101]**.	To record the charges as one-time purchase fees in your general ledger.
Click **S for Sponsorable**.	To enable the rate to be paid by another account. Select this option to enable one account to pay for another.
Click **D** for **Discountable**.	To enable the rate to be discounted when the service is purchased.
In the **Fixed Amount** field, type **15**.	One-time charges, such as the purchase fee in this product, are called “fixed" rates. You enter the dollar amount of the charge.
In **Units**, select **None [0]**.	This rate uses the **Occurrence** ratable usage metric (RUM), which is not measured in units.
Click **OK**.	This returns you to the Product Attributes dialog box. You are finished creating purchase fees. Now you must create the cycle fees.

Product 2a - Creating the Cycle Fees

You create the cycle fees by defining two rate tiers that map to the cycle forward monthly event. The first rate tier gives the customer the third month free, and the second rate tier charges the basic monthly fee of $19.95.

Product 2a - Creating the Free Month

You define the free month rate tier first to ensure that the free month rate is applied first instead of the standard monthly rate. You define the free month as a certain number of days from enrollment.

Table 12-9 describes the steps for creating the free month.

Table 12-9 Creating the Free Month

Do This	Why
In the **Event Map** group, click **Add**. In the **Event** column, select **Monthly Cycle Forward Event**.	To create the sign-up fee by associating the rate details with the purchase event, as specified by the plan.
In the **Measured By** column, select **Occurrence**. See ratable usage metric (RUM).	To specify that the fee is charged when the monthly cycle forward event occurs.
In the **Rate Plan Structure** column, select **Single Rate Plan**. Then, under **Single Rate Plan**, click **Open Rate Plan**.	To set up the rates for this product.
In the **Plan Detail** tab, enter **cycle_forward_monthly** in the **Name** field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose.
In the **Rate Structure** tree view, select **Tier 1**. In the **Tier Detail** tab, enter a name for the rate tier: **Third Month Free**.	The name identifies the rate tier in the product and reminds you of when it is valid.
Under **Effective Dates**, select **Relative**.	To make the rate available for a specific period relative to the product purchase date.
Under **Starts**, enter **55** and **Day**.	To specify that this rate starts 55 days after the product is purchased. After the customer pays for the first two months, BRM runs billing for the third month 59-62 days after registration. You want BRM to use the free month rate, so by specifying 55 days, you ensure that this rate is applied before BRM runs billing for the customer's third month.
Under **Ends**, enter **65** and **Day**.	To specify that this rate ends 65 days after the product is purchased. This is after BRM runs billing for the third month.
Under **Use Time/Day Restrictions** select **No restrictions**.	To bypass setting valid day ranges.
In the **Rate Structure tree view,** select **Rate 1. In the Rate Data tab,** enter a name for the rate: **No Monthly Fee**.	To identify the purpose of this rate tier.
Select **Overrides credit limit**.	To continue to bill for the email account if the customer exceeds the credit limit.
In the **Based on** list, select **Continuous**.	This option is the default when you use only one quantity discount bracket to rate an event.

Table 12-9 (Cont.) Creating the Free Month

Do This	Why
In the **Rate Structure** tree view under the **No Monthly Fee** rate, select **No Minimum - No Maximum**.	This displays the **Balance Impacts** tab, where you define the cost of the cycle rate.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the impacts list and to specify the balance impacts **-** what you charge the customer for the service.
Click the **Resource ID** box and scroll to select **US Dollar [840]**.	To bill in US dollars.
In the **GLID** list, select **Monthly Fees [102]**.	To have the free month recorded as monthly fees in a general ledger accounting report.
For **Fixed** and **Scaled**, make sure the values are **0**.	You do not charge the customer for the third month.
In the **Rate Structure** tree view, click the **cycle_forward_monthly** rate plan.	You are finished creating this rate tier, and now must create the standard monthly fee.

Product 2a - Creating the Standard Monthly Fee

You define another rate tier to charge the $19.95 monthly fee. You create this rate tier second to ensure that the “free month" rate is applied first and instead of the standard monthly rate that one time.

Table 12-10 describes the steps for creating the standard monthly fee.

Table 12-10 Creating the Standard Monthly Fee

Do This	Why
In the **Rate Structure** tree view, select the **cycle_forward_monthly rate plan** and click **Add**.	To add a new rate tier to the existing cycle_forward_monthly rate plan.
In the **Tier Detail** tab, enter a name for the rate tier: **Monthly Access Fee**.	The name identifies the rate tier in the product and reminds you of when its valid.
Under **Tier Detail**, select **Absolute** or **Relative**, then **Starts Immediately** and **Never Ends**. Also, under **Use Time/Day Restrictions**, select **No restrictions**.	To make the rate always available and always charged when no other rates are valid.
In the **Rate Structure tree view,** select **New Rate. In the Rate Data tab,** enter a name for the rate: **$19.95 a Month**.	To identify the fee in this rate tier.
Select **Overrides credit limit**.	To continue to charge for the email account when the customer exceeds the credit limit.
In the **Based on** list, select **Continuous**.	This option is the default when you use only one quantity discount bracket to rate an event.
In the **Rate Structure tree view,** select **No Minimum - No Maximum.**	This displays the **Balance Impacts** tab, where you define the cost of the cycle rate.

Table 12-10 (Cont.) Creating the Standard Monthly Fee

Do This	Why
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Under **Balance Impacts**, click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the impacts list and to specify the balance impacts - what you charge the customer for the service.
Click the **Resource ID** box and scroll to select **US Dollar [840]**.	To bill in US dollars.
In the **GLID** list, select **Monthly Fees [102]**.	To have the free month recorded as monthly fees in a general ledger accounting report.
In **Scaled Amount**, enter **19.95**.	Recurring fees, such as the monthly fee in this product, are called *scaled* rates. You enter the dollar amount of the fee.
Click **OK**.	You are finished creating this product.

Product 2b – Email Account, No Fees

The purpose of this product is to provide the customer with a single email account. Since the email account is included in the standard monthly fees for this plan, you do not charge the customer in this product.

Product 2b - Creating the Product

Table 12-11 describes the steps for creating the email account product.

Table 12-11 Creating the Product

Do This	Why
In Pricing Center, choose **Insert – Product** and select **Create a new product**.	To create a product.
Click **Next** and give the product a name: **Product 2b – Email Account, No Fees**. (You can also add a description.)	The product name identifies the product in Pricing Center. This name must be unique to all products in the same database brand.
When asked what type of product to create, select **Subscription** and in **Applies To**, choose **/service/email**.	This enables the product to contain recurring monthly rates and associates it with the email service.
Continue through the product creation wizard until you get to the **Valid Purchase Quantity** group.	Although you can change various attributes of the product while using the wizard, this sample product uses the default attributes up to this point.
Click **Maximum of** and enter **1**.	To specify that a customer can purchase no more than one email account.
Finish the product creation wizard, and click **Yes** at the end.	After you finish the wizard, you specify the rates for this product.

Product 2b - Creating the Free Rate

Even though the email account is free, you still must define an event rating map to track the free email service each month. You map the free email “fee" to the monthly cycle event.

Table 12-12 describes the steps for creating the free rate.

Table 12-12 Creating the Free Rate

Do This	Why
In the **Event Map** group, click **Add**. In the **Event Type** column, select **Monthly Cycle Forward Event**	To create the free monthly access fee by associating the rate details with the cycle event, as specified by the plan.
In the **Measured By column**, select the **Occurrence** ratable usage metric (RUM).	To specify that the rate is charged every month when the cycle forward event occurs. **Note:** The rating details are only used for time-dependent fees.
In the **Rate Plan Structure** column, select **Single Rate Plan.** Under **Single Rate Plan**, click **Open Rate Plan**.	To set up the rates for this product.
In the **Plan Detail** tab, enter **cycle_forward_monthly** in the **Name** field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose. For system events, you must enter the predefined name of the rate plan.
In the **Rate Structure** tree view, select **Tier 1**. In the **Tier Detail** tab, enter a name for the rate tier: **Monthly Email Fee**.	The name identifies the rate tier in the product and reminds you when it is valid.
Under **Effective Dates**, select **Absolute**.	To set an indefinite period when the rate is available. **Note:** do not use **Always Applies**, because that sets a rate as a default rate, and charges the rate when no other rates apply. Since this rate is free, you do not want it to be the default rate.
Under **Use Time/Day Restrictions**, select **No restrictions**.	This rate is always valid; therefore, you do not need to specify which days it is valid.
In the **Rate Structure** tree view**,** select **Rate 1. In the Rate Data tab,** enter a name for the rate: **No Charges**. **Note:** Rate names must be unique.	To remind you of the purpose of this rate.
In the **Based on** list, select **Continuous**.	This option is the default when you use only one quantity discount bracket to rate an event.
In the **Rate Structure** tree view**,** select **No Minimum- No Maximum to display the Balance Impacts** tab.	This displays the **Balance Impacts** tab, where you define the cost of the rate.
Under **Valid Quantities**, select **None** for both the **Minimum** and **Maximum** fields.	The monthly fee is not based on a specific amount of the event.
Under **Balance Impacts**, click **Add**. **Tip:** Drag the right edge of any column heading to make the list wider.	To add a row to the impacts list and to specify the balance impacts **-** what you charge the customer for the service.
Click the **Resource ID** box and scroll to select **US Dollar [840]**.	To bill in US dollars.
In the **GLID** list, select **Monthly Fees [102]**.	To record the charges as monthly fees in your general ledger.
For **Fixed** and **Scaled**, make sure the values are **0**.	You do not charge the customer for the third month.
In **Units**, select **None [0]**.	This rate uses the **Occurrence** ratable usage metric (RUM), which is not measured in units.

Table 12-12 (Cont.) Creating the Free Rate

Do This	Why
Click **OK**.	You are finished creating the email product.

Deal 2a – Unlimited Internet Service

This deal contains the product for the sign-up fee and the Internet usage fees.

1. In Pricing Center, choose **Insert – Deal**.
2. Name the deal: **Deal 2a – Unlimited Internet Service**. (You can also add a description.)
3. In the **Applies to** list, select **/service/ip**.
4. In the **Products Associated with Deal** group, click **Add**.
5. In the **Product Name** list, select **Product 2a – Internet Access with Free Period**.
6. Click **OK** twice.

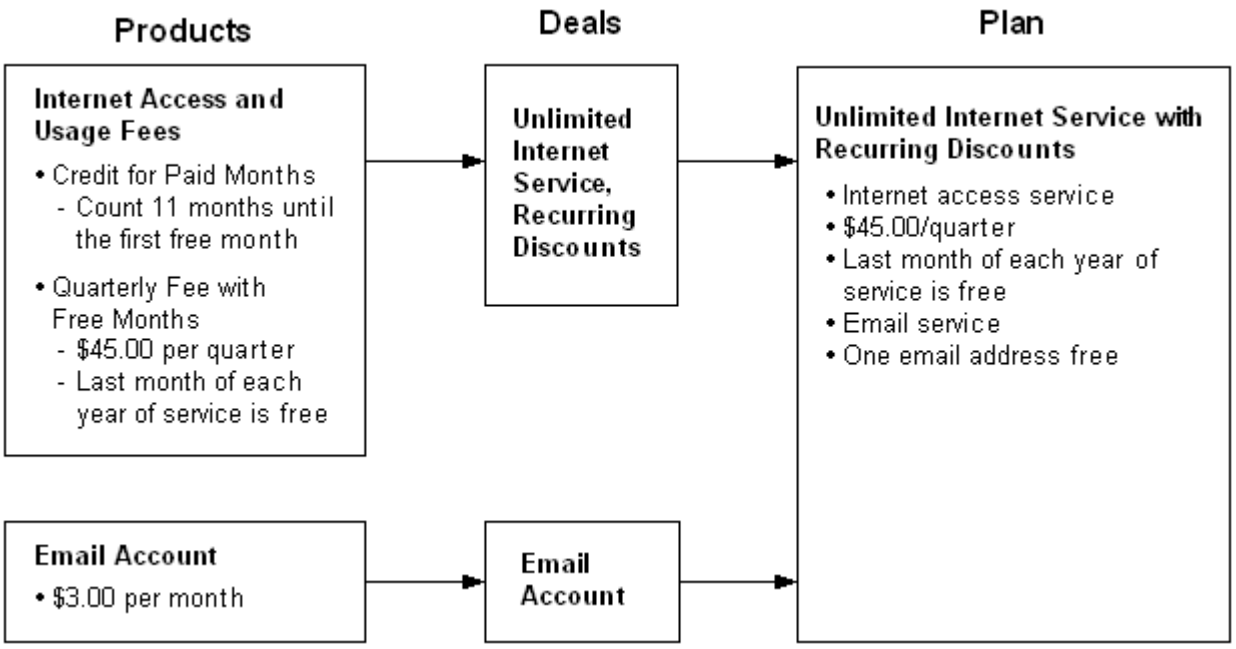
Deal 2b – Discounted Email Access

1. In Pricing Center, choose **Insert – Deal**.
2. Name the deal: **Deal 2b – Discounted Email Access**. (You can also add a description.)
3. In the **Applies to** list, select **service/email**.
4. In the **Products Associated with Deal** group, click **Add**.
5. In the **Product Name** list, select **Product 2b – Email Account, No Fees**.
6. Click **OK** twice.

Re-Creating Plan 3 – Unlimited Internet Service with Recurring Discounts

Figure 12-4 shows how to implement Internet service with recurring discounts.

Figure 12-4 Unlimited Internet Service with Recurring Discounts Plan



Putting together Plan 3

1. Create a custom BRM resource to use with this plan.

   BRM *resources* are units of exchange, such as dollars or hours, that are charged or credited when a customer uses your service. BRM comes with several common resources that you can use. For this plan, you must create one additional resource called “Paid Quarter". See "Creating a custom resource for Plan 3".

2. Create a product to charge for the Internet service based on which quarter of the year it is. See "Product 3a – Prepaid Internet Service with Free Period".

   > **Note:**
   >
   > To offer the email account, you could create a second product specifically for Plan 3. However, Plan 2 has a product that already defines the email rate you need. You can use that product as a component of this plan, too.

3. Create a deal for Internet access. See "Deal 3 – Unlimited Internet Service, Recurring Discounts".
4. In Pricing Center, choose **Insert – Plan**.
5. Name the plan: **Plan 3 – Unlimited Web Access with Recurring Discounts**. (You can also add a description.)
6. In the **Services** section, click **Add**.
7. For **Service Type**, select **/service/ip**.
8. For Deal, select **Deal 3a – Unlimited Internet Service, Recurring Discounts**.
9. Click **OK**.

10. In the **Services** section, click **Add**.
11. For **Service Type**, select **/service/email**.
12. For **Deal**, select **Deal 2b – Discounted Email Access**.

    As with products, you also can use a deal from another plan as a building block in this plan.
13. Click **OK**.
14. In the **Credit Limits** section, click **Add**.
15. In the **Resource ID** list, select **Paid Quarter**.
16. In the **Credit Limit** field, de-select **None**, then type **3**.
17. Click **OK**.
18. Click **OK** again to finish the plan.

Creating a custom resource for Plan 3

Table 12-13 describes the steps for creating a custom resource.

Table 12-13 Creating a Custom Resource

Do This	Why
From the start options, choose **Pricing Center**, which is under **Portal**, and then supply the login name and password to open the Pricing Center application.	You create resources with the Resource Editor. **Important:** Check with your BRM system administrator before adding resources to the BRM database.
Click **Resource Editor**.	You use the Editor to add new resources and modify existing resources.
Choose **Edit – Add New Item**.	To add the resource required by this plan.
In the **Resource ID** box, type **1000012**.	This number identifies the resource in BRM. When creating a resource, you can use any unique number between 1000001 and 4000000000. (Numbers below 1000001 are reserved for use by BRM.)
In the **Name** box, type **Paid Quarter**.	The name appears in the resource list in Pricing Center.
In the **Round To** box, select **0**.	BRM uses a whole number for the free month, so you do not need any rounding.
For **Status**, select **Active**.	BRM can only use active resources.
Click **OK**.	You are finished creating this resource.
Choose **File – Close** and **Commit New Changes**.	To add the new resource to the BRM database and close the Resource ID Editor.
Click **Yes** to confirm.	To confirm that you want to add the resource.

Product 3a – Prepaid Internet Service with Free Period

This product serves several purposes:

- Charging the customer for 3 months of service at the beginning of each quarter.
- Counting the number of quarters the customer has been with the plan by tracking a “paid quarter" for each quarter. This is the resource you created for this plan.

- When the customer has remained with the plan for 3 quarters, this product credits the customer with $15 (the cost of one month of service) and resets the “paid quarter" resource to 0 to start the countdown for the next year.

Product 3a - Defining the Product

Table 12-14 describes the steps for defining the prepaid internet service with free period product.

Table 12-14 Defining Prepaid Internet Service with Free Period

Do This	Why
In Pricing Center, choose **Insert** – **Product** and select **Create a new product**.	To create a product.
Click **Next** and give the product a name: **Product 3a – Prepaid Internet Service with Free Period**. (You can also add a description.)	The product name identifies the product in Pricing Center. This name must be unique to all products in the same database brand.
Click **Next**. When asked what type of product to create, select **Subscription** and in **Applies To**, choose **/service/ip**.	This enables the product to contain usage rates and recurring monthly rates and associates it with the IP service.
Continue through the rest of the product creation wizard, and click **Yes** at the end.	Although you can change other attributes of the product while using the wizard, this sample product uses the remaining default attributes.

Product 3a – Creating the Quarterly Fee

This event rating map charges the customer for 3 months of service at the beginning of each quarter, and discounts the third quarterly fee by $15 to provide a free month of service.

It provides the discount by tracking one paid quarter for each quarter the customer has been with the plan. Then, when the customer reaches the third quarter, the rate is discounted $15 (the cost of one month of service) and the paid quarter balance is reset to 0 to start the countdown for the next year.

The rate uses event quantity ranges to determine the correct fee.

Table 12-15 describes the steps for determining the quarterly fee.

Table 12-15 Determining Fee

Do This	Why
In the **Event Map** group, click **Add**. In the **Event** column, select **Quarterly Cycle Forward Event**.	To specify billing at the beginning of each three-month cycle, as specified by the plan.
In the **Measured By column**, select **Occurrence**. See ratable usage metric (RUM).	To specify that the rate is charged every three months when the quarterly cycle forward event occurs. **Note**: The rating details are used only for time-dependent fees.
In the **Rate Plan Structure** column, select **Single Rate Plan**. Under **Single Rate Plan**, click **Open Rate Plan**.	To set up the rates for this product.

Table 12-15 (Cont.) Determining Fee

Do This	Why
In the **Plan Detail** tab, enter **cycle_forward_quarterly** in the Name field.	The rate plan name identifies the rate plan in the product and reminds you of its purpose. For system events, you must enter the predefined name of the rate plan.

Deal 3 – Unlimited Internet Service, Recurring Discounts

This deal bundles the product for the monthly counter, the product for the quarterly fees and free month, and the product for the email account.

1. In Pricing Center, choose **Insert – Deal**.
2. Name the deal: **Deal 3 – Unlimited Internet Service, Recurring Discounts**. (You can also add a description.)
3. In the **Applies to** list, select **/service/ip**.
4. In the **Products Associated with Deal** group, click **Add**.
5. In the **Product Name** list, select **Product 3a – Prepaid Internet Service with Free Period**.
6. Click **OK**.

Descriptions of the Advanced Sample Plans

This section describes each of the advanced sample plans including why certain pricing strategies were used to construct each plan. Each plan includes a description of what the plan does, key concepts illustrated by the plan, the customizations required to implement the plan, the structure of the plan, the target customer, and questions about the plan's structure and design.

ASP User Profile Plan

This plan (**ASPUserProfile.IPL)** provides an Enterprise Resource Planning (ERP) product with three categories of users: active, casual, and self-service. It charges $500 per month per active user, $150 per month per casual user, and $5 per month per self-service user. This plan charges a $22,000 minimum. After the per user charges reach $22,000, they begin to affect the customer's bill.

In addition, this plan provides volume discounts. If a customer is charged for more than $35,000 in a month, the charge for that month is discounted 10%. Similarly, if a customer is charged more than $40,000, the discount is 15%. If a customer is charged more than $50,000, the discount is 20%.

Concepts Illustrated

Real-time volume discounting and minimum charge.

Customization Required for the ASP User Profile Plan

The ASP User Profile plan includes the following custom resources:

- ERP Active (resource ID = 1000022)
- ERP Casual (resource ID = 1000023)
- ERP SelfService (resource ID = 1000024)
- Pre-discount Dollars (resource ID = 1000025)

You must add these custom resources to your BRM database if you intend to use the plan.

Adding custom resources

To add the custom resources:

1. Start the Configuration Center and connect to your BRM database.
2. Ensure that your database does not include custom resources with any of the resource IDs above: 1000022 through 10000025.

   If your database does include custom resources with those resource IDs, you must create the custom resources with different resource IDs and then change the plan to refer to those new resource IDs.
3. Add the resources.

Structure of the ASP User Profile Plan

The plan contains one product, ERP Product and six rate plans, add_active_user, add_casual_user, add_selfservice_user, del_active_user, del_casual_user, del_selfservice_user. In addition, this product contains one Fold rate plan selector that contains two rate plans, Fold Pre-discount dollars to US Dollars and Fold Royalty.

add_active_user rate plan

For each active user added, this rate plan increments the Pre-discount dollars resource by $500 and the ERP Active resource by 1.

add_casual_user rate plan

For each casual user added, this rate plan increments the Pre-discount dollars resource by $150 and the ERP Casual resource by 1.

add_selfservice_user rate plan

For each self-service user added, this rate plan increments the Pre-discount dollars resource by $5 and the ERP SelfService resource by 1

del_active_user rate plan

For each active user deleted, this rate plan decrements the Pre-discount dollars resource by $500 and the ERP active resource by 1.

del_casual_user rate plan

For each casual user deleted, this rate plan decrements the Pre-discount dollars resource by $150 and the ERP casual resource by 1.

Fold rate plan selector

This rate plan selector contains two rate plans: **Fold Pre-Discount Dollars to US Dollars** and **Fold Royalty**.

The **Fold Pre-Discount Dollars to US Dollars** rate plan folds Pre-Discount Dollars into US Dollars resource balance based on the Pre-Discount Dollars resource balance. This rate plan contains five quantity discount brackets:

- The first quantity discount bracket increments the US Dollars resource by 22,000 if the Pre-Discount Dollars resource balance is under 22,000. This is, in effect, a minimum charge.
- The second quantity discount bracket applies if the balance of Pre-Discount Dollars is between 20,001 and 35,000. It multiplies the Pre-Discount Dollars resource balance by 1.00 and folds it into US Dollars.
- The third quantity discount bracket applies if the balance of Pre-Discount Dollars is between 35,001 and 40,000. It multiplies the Pre-Discount Dollar resource balance by 0.9 and folds it into the US Dollars resource balance. This is, in effect, a 10% discount for spending more than 35,000 Pre-Discount Dollars.
- The fourth quantity discount bracket applies if the balance of Pre-Discount Dollars is between 40,001 and 50,000. It multiplies the Pre-Discount Dollar resource balance by 0.85 and folds it into the US Dollars resource balance. This is, in effect, a 15% discount for spending more than 40,000 Pre-Discount Dollars.
- The fifth quantity discount bracket applies if the balance of Pre-Discount Dollars is greater than 50,000. It multiplies the Pre-Discount Dollar resource balance by 0.8 and folds it into the US Dollars resource balance. This is, in effect, a 20% discount for spending more than 50,000 Pre-Discount Dollars.

Questions about the Plan

Why is the Pre-Discount Dollars custom resource necessary?

The Pre-Discount Dollars custom resource is necessary to implement a minimum monthly charge and volume discounting. You track Pre-Discount Dollars instead and fold them into US Dollars at the end of the month to determine how the customer should be charged and which, if any, level of discounting should be given.

Audio/Video Plan

This plan (**AudioVideo.IPL)** gives 120 minutes of audio streaming and 60 minutes of video streaming per month for $14.95. The first month is free. Any audio minutes or video minutes remaining at the end of a month are lost.

After the first 120 minutes of audio streaming and the first 60 minutes of video streaming used in a month the plan charges for additional minutes based on time of day. From 8:30 a.m. to 6:00 p.m., audio is 25 cents a minute and video is 35 cents a minute. From 6:00 p.m. to 8:30 a.m., audio is 15 cents a minute and video is 25 cents a minute.

Concepts Illustrated

First month free, peak and off-peak hours, custom resources.

Customization Required for the Audio/Video Plan

The Audio/Video plan includes two custom resources:

- Minute Streamed Audio (Resource ID = 1000005)
- Minute Streamed Video (Resource ID = 1000006)

You must add these custom resources to your BRM database if you intend to use the plan.

Adding custom resources

To add the custom resources:

1. Start the Resource Editor and connect to your BRM database.
2. Ensure that your database does not already include custom resources with any of these Resource IDs: 1000005 or 10000006.

   If your database does include custom resources with those Resource IDs, you must create the custom resources with different Resource IDs and then change the plan to refer to those new Resource IDs.
3. Add the resources.

Structure of the Audio/Video Plan

The plan contains three products: AudioVideo Monthly, AudioVideo Audio Usage, and AudioVideo Video Usage.

AudioVideo Monthly product

This product contains four rate plans. Two are single rate plans and two are combined in a rate plan selector.

The Monthly Cycle Forward Event rate plan contains an AudioVideo Monthly Rate Tier that charges $14.95 per month for the plan.

The Product Purchase Fee Event rate plan contains a Purchase Rate Tier that discounts the first month $14.95 when the plan is purchased. In effect, this makes the first month of service free.

The Cycle Fold Event rate plan selector combines two rate plans:

- The first rate plan, called Audio Fold, contains an Audio Fold Rate Tier that multiplies the balance of the Minute Streamed Audio resource by negative one (-1) and adds it to the resource balance. This resets the Minute Streamed Audio resource balance to zero.
- The second rate plan, called Video Fold, contains a Video Fold Rate Tier that multiplies the balance of the Minute Streamed Video resource by negative one (-1) and adds it to the resource balance. This resets the Minute Streamed Video resource balance to zero.

AudioVideo Audio Usage

This product contains one rate plan, IP Audio Event, with two rate tiers, Audio Basic Minutes and Audio Excess Minutes:

- The Audio Basic Minutes tier charges nothing for each minute of audio streamed up to 120 minutes per month.
- The Audio Excess Minutes tier charges a variable rate for each minute of audio streamed over 120 minutes per month. The rate structure includes two time of day periods: Peak and Off Peak. During Peak hours, 8:30 a.m. to 6:00 p.m., streamed audio is charged at 25 cents a minute. During Off Peak hours, 6:00 p.m. to 8:30 a.m., streamed audio is charged at 15 cents a minute.

AudioVideo Video Usage

This product contains one rate plan, IP Video Event, with two rate tiers, Video Basic Minutes and Video Excess Minutes:

- The Video Basic Minutes tier charges nothing for each minute of video streamed up to 60 minutes per month.
- The Video Excess Minutes tier charges a variable rate for each minute of video streamed over 60 minutes per month. The rate structure includes two time-of-day periods: Peak and Off Peak. During Peak hours, 8:30 a.m. to 6:00 p.m., streamed video is charged at 35 cents a minute. During Off Peak hours, 6:00 p.m. to 8:30 a.m., streamed video is charged at 25 cents a minute.

Questions about the Plan

Why not just delay the initial monthly charge for one month instead of including a purchase product that refunds the first month's charge?

By including a purchase product that refunds the first month's charge, you can quickly change the plan to offer only a partial refund in the first month.

Customer Service Plan

This plan (**CustomerService.IPL)** is designed to let one company offer hosted customer service applications to a second company. The second company is then charged a fee for each subscriber to the service and is also charged for customer service minutes used by their subscribers.

The second company is charged a fee per subscriber and is also given a certain amount of customer service minutes per subscriber based on the total number of subscribers the second company has. The number of subscribers is checked when the Customer Service plan is first purchased and each time BRM billing is run - typically at the end of each month.

The charge per subscriber and the number of free customer service minutes given is determined as shown in Table 12-16:

Table 12-16 Subscribers and Charges

Number of Subscribers	Charge per Subscriber	Number of Free Customer Service Minutes per Subscriber
0 to 5000	$5	2
5000 to 10000	$4	3
10,000+	$3	4

Concepts Illustrated

Tiered pricing, products supporting business to business scenarios.

Customization Required for the Customer Service Plan

The Customer Service plan includes one custom service:

- **/service/customerservice**

The Customer Service plan includes two different custom resources:

- Customer Service Minute (resource ID = 1000026)
- Subscription (resource ID = 1000502)

The Customer Service plan also includes two custom events:

- **/event/activity/add_subscription**
- **/event/activity/delete_subscription**

You must add these custom services, resources, and events to your BRM database if you intend to use the plan.

Adding custom services

You must use the Developer Workshop to add the **/service/customerservice** custom service to your BRM database.

Adding custom resources

To add the custom resources:

1. Start the Resource Editor and connect to your BRM database.
2. Ensure that your database does not include custom resources with any of the resource IDs above: 1000026 or 1000502.

   If your database does include custom resources with those resource IDs, you must create the custom resources with different resource IDs and then change the plan to refer to those new resource IDs.
3. Add the resources.

Adding custom events

This plan includes two custom events: **/event/activity/add_subscription** and **/event/activity/delete_subscription**.

You must use the Developer Workshop and edit **pin_event_map** to add these custom events to your BRM database.

Structure of the Customer Service Plan

The plan contains one product, Customer Service Product, with four rate plans, Add Subscription Event, Delete Subscription Event, Monthly Cycle Forward Event, Session Event, and one rate plan selector, Cycle Fold Event.

Add Subscription Event

This rate plan adds one to the Subscription resource when an **/event/activity/add_subscription** event occurs.

Delete Subscription Event

This rate plan deletes one from the Subscription resource when an **/event/activity/delete_resource** event occurs.

Monthly Cycle Forward Event

This rate plan has a customer service rate tier that both charges per subscription and allots free minutes based on the number of total subscriptions as shown in Table 12-17:

Table 12-17 Subscribers and Charges for Monthly Cycle Forward Event

Number of Subscribers	Charge per Subscriber	Number of Free Customer Service Minutes per Subscriber
0 to 5000	$5	2 (negative 2 added to the resource balance)
5000 to 10000	$4	3 (negative 3 added to the resource balance)
10,000+	$3	4 (negative 4 added to the resource balance)

Session Event

This rate plan has one rate tier, the Customer Service Usage Rate Tier, that adds one to the Customer Service Minute resource balance for each customer service minute used by the company's subscribers.

Cycle Fold Event

This rate plan selector has one rate plan, Fold Minutes. The rate plan folds the Customer Service Minute resource balance to US Dollars, $1 for each 1 minute used over the minutes given by the Cycle Forward Monthly Event rate plan. Then it folds the Customer Service Minute resource balance to zero by multiplying the balance by negative 1 and then adding it to the balance.

Note that the order in which the folds occur is very important. If the Customer Service Minute resource balance is folded first before the number of US Dollars is calculated, the company will never be charged.

Questions about the Plan

What happens if the number of subscribers during a month increases enough to qualify for a lower rate per subscriber and more customer service minutes per subscriber?

The rate tier is only checked when the plan is first purchased and each month when billing is run. For example, if a company starts a month with 4500 subscribers and then adds 700 subscribers during the month, they will still be charged per subscriber and given customer service minutes per subscriber based on having 4500 subscribers until the company runs billing again.

If all of the customer service minutes given to a company for the number of subscribers they have are not used by the end of the month, why does not the fold cause a credit to be issued to US Dollars?

The fold quantity bracket starts at “0" with no maximum so it is not activated if the customer service minute balance is negative. If you actually wanted a credit to be issued for customer service minutes not used, you could add another bracket to the fold that starts at no minimum and ends at “0".

IP Limited Access Plan

This plan (**IPLimitedAccess.IPL**) gives 100 hours of IP access for $19.95 per month. Each hour of IP access after a 100 hours is charged at $2.00 an hour. The first 12th month of IP service is free. Unused hours in each month are lost.

For every IP access hour used, the customer gets 1 frequent flier credit at the end of the billing period. In addition, for every IP access hour in excess of 100 in a month, the customer is given 1 game credit which must be used by the end of that month.

The plan also includes unlimited email service for a one-time purchase fee of $5.00.

Concepts Illustrated

Free hours, free 12th month, hourly credits, folds

Customization Required for the IP Limited Access Plan

The IP Limited Access plan includes several different custom resources:

- Bulk hours (Resource ID = 1000013)
- paid month (Resource ID = 1000008)
- Frequent Flier Miles (Resource ID = 1000003)
- Free Game (Resource ID = 1000004)

You must add these custom resources to your BRM database if you intend to use the plan.

Adding custom resources

To add the custom resources:

1. Start the Resource Editor and connect to your BRM database.
2. Ensure that your database does not include custom resources with any of the Resource IDs above: 1000003, 1000004, 1000008, or 1000013.

   If your database does include custom resources with those Resource IDs, you must create the custom resources with different Resource IDs and then change the plan to refer to those new Resource IDs.
3. Add the resources.

Structure of the IP Limited Access Plan

The plan contains three products: IP Limited Access Email, IP Limited Access Monthly, and IP Limited Access Usage.

IP Limited Access Email product

This is an item product consisting of a one-time charge of $5.00 for unlimited email service.

IP Limited Access Monthly product

This is a subscription product containing two event types: Monthly Cycle Forward Event and Cycle Fold Event.

Monthly Cycle Forward Event

This includes a monthly rate tier with three quantity discount brackets.

- The first bracket charges $19.95 per month and increments the Paid Month resource by 1 each month.
- When Paid Months reaches 11, the second quantity discount bracket takes effect. The second bracket increments the Paid Month Resource by 1 but does not charge for that

month. This bracket is valid only when Paid Month is equal to or greater than 11 and less than 12, so it gives the customer the 12th month of service free.

- The third quantity bracket takes effect after the 12th month of service. Since this bracket is valid when Paid Month is equal to or greater than 12, it remains in effect indefinitely after the 12th month of service. It affects resources the same way the first bracket does, thereby ensuring that only the first 12th month rather than every 12th month of service is free.

Cycle Fold Event

The Cycle Fold Event includes two fold rates which zero out the game credit balance at the end of the month and any hours of IP usage accrued in the month.

The folds zero out any remaining game credits and hours of IP service by multiplying the appropriate resource balance by -1 and adding it to the resource balance. For example, if a customer used 35 Bulk Hours of IP service and has 3 game credits left at the end of a month, the respective resource balances are 35 and 3. The fold rates multiply those remaining resource values by -1, resulting in -35 and -3 respectively, and adds those results to the resource balances to end with (35 + negative 35) or 0 Bulk hours and (3 + negative 3) or 0 Game Credits.

IP Limited Access Usage product

This is a subscription product that includes one event type: Session Event.

Session Event includes two rate tiers.

- Base Rate Tier

  This rate tier is in effect until the Bulk Hours resource for a month reaches “100.00".

  In this tier, for each hour of service, the Bulk Hours resource is incremented by 1 and the Frequent Flier miles resource is incremented by 1. Thus, each hour decreases the number of base IP hours available and adds 1 frequent flier mile to the customer's account. The hours in the Base Rate Tier, the first 100 hours of IP usage in a month, have a balance impact of “0" because there is no usage fee for the first 100 hours in a month.

- Excess Rate Tier

  This rate tier takes effect after the first 100 hours of IP usage in a month.

  In this tier, for each hour of service, the customer is charged $2 and given 1 frequent flier mile and 1 game credit.

Questions about the Plan

How do I make every 12th month of base 100 hours of IP service free instead of just the first 12th month?

To make every 12th month free instead of just the first 12th month, change the quantity discount brackets within the Monthly Cycle Forward Event of the IP Limited Access Monthly product:

1. Leave the first quantity discount bracket as is.
2. Delete the third bracket.
3. Change the second quantity discount bracket as follows:
   a. Change the Maximum Valid Quantities to None.
   b. Change the Paid Month resource impact from 1.00 to (11.00).

This zeroes out the Paid Month resource so the first bracket is used until the next 12th month of service is reached.

Since the number of Bulk Hours a customer uses is irrelevant as far as fees are concerned once they use over 100 in a month, why does the Excess Rate tier of IP Limited Access Usage continue to track the number of Bulk Hours used?

If Bulk Hours were not tracked by the Excess Rate tier, CSRs and customers would have no easy way of knowing how many excess hours the customer was charged for in a month. This way, the exact number of excess hours appears in Customer Center.

If Bulk Hours were not tracked, CSRs and customers must divide the excess charges by $2 to find the number of excess hours used in a month. But if excess hours were charged on a sliding scale by the plan, say $2 for the first 50 excess hours, $1.50 for the next 50 excess hours, and $.75 thereafter, the calculation would become more difficult and time consuming to perform. This way, the number of excess hours used is immediately available.

Online Articles Plan

This plan (**On-LineArticles.IPL)** charges $30.00 for quarterly access to an online articles service. The first five articles downloaded per month are free. The second five articles downloaded are charged at $2.00 an article. The next ten articles downloaded are charged at $1.50 an article. After that, all articles are charged at $1.00 an article.

In addition, if a customer downloads 30 or more articles in a month, the entire usage charge for that month is discounted 15%.

Concepts Illustrated

Quarterly cycle fee, variable usage fees based on monthly volume, discounting of all usage fees when a particular level of usage is reached.

Customization Required for the Online Articles Plan

The Online Articles plan includes two custom resources:

- Num Articles (resource ID = 1000015)
- Usage Charge (resource ID = 1000014)

You must add these custom resources to your BRM database if you intend to use the plan.

Adding custom resources

To add the custom resources:

1. Start the Resource Editor and connect to your BRM database.
2. Ensure that your database does not include custom resources with any of the resource IDs above: 1000014 or 10000015.

   If your database does include custom resources with those resource IDs, you must create the custom resources with different resource IDs and then change the plan to refer to those new resource IDs.
3. Add the resources.

Structure of the Online Articles Plan

The plan contains one product, On-Line Articles; two rate plans, Quarterly Cycle Forward Event, and Content Download Event; and one rate plan selector, Cycle Fold Event.

Quarterly Cycle Forward Event rate plan

This rate plan charges $30.00 per quarter. Note that the rate plan name is **cycle_fold_quarterly**.

Content Download Event rate plan

This rate plan includes four quantity discount brackets:

- The first quantity discount bracket increments the Num Articles resource by 1 for each article downloaded per month until 5 articles are downloaded.
- The second quantity discount bracket increments the Num Articles resource by 1 and increments the Usage Charge resource by 2.00 for each article downloaded per month until 10 articles are downloaded.
- The third quantity discount bracket increments the Num Articles resource by 1 and increments the Usage Charge by 1.50 for each article downloaded per month until 20 articles are downloaded.
- The fourth quantity discount bracket increments the Num Articles resource by 1 and increments the Usage Charge by 1.00 for each article downloaded over 20 per month with no maximum.

Cycle Fold Event rate plan selector

This rate plan selector contains two rate plans:

- The first rate plan, Fold Usage Rate Plan, folds the Usage Charge into US Dollars based on the number of articles downloaded. If the Num Articles resource is less than 30, the Usage Charge resource balance is multiplied by 1.00 and folded directly into US Dollars. If the Num Articles resource is 30 or greater, the Usage Charge resource balance is multiplied by .85 and then folded into US Dollars. This, in effect, gives the user a 15% discount on all of their usage charges if they download 30 or more articles in a month.
- The second rate plan, Fold Articles Rate Plan, folds the Num Articles resource to zero by multiplying the balance by negative 1 and adding it to the balance.

Note that the order in which these rate plans are applied is very important. If the Fold Articles Rate Plan is applied first, customers will never get a 15% discount on their usage regardless of how many articles they download in a month.

You can control the order in which fold rate plans are applied by making sure the resource that you want to fold first either has a lower resource ID than the other resources you want to fold or is listed above the other resources you want to fold in the **pin_beid** file.

Questions about the Plan

Why is the Usage Charge custom resource necessary?

The Usage Charge custom resource is necessary because the 15% discount for 30 or more articles downloaded a month applies only to usage charges. Therefore you need a way of tracking usage fees separately from any other fees. If you simply charged US Dollars in the Content Download Event rate plan, you must either discount all customer fees, including any quarterly fees, or not discount customer fees at all.

T1Access Plan

This plan (**T1Access.IPL)** includes a custom T1 access service. It provides the service for a monthly charge of $5,000 and an initial setup fee of $600. In addition, if the customer cancels the service before a year after the purchase date, they are charged a $10,000 cancellation fee.

Concepts Illustrated

Cancellation fee, custom service, relative date ranges for tiers

Customization Required for the T1 Access Plan

The T1 Access plan includes the custom service **service/T1.**

You must use the Developer Workshop to add this custom service to your BRM database.

Structure of the T1 Access Plan

The plan contains three products: T1 Purchase, T1 Monthly, and T1 Cancellation.

T1 Purchase product

This product consists of a one-time purchase fee of $600 for starting the T1 service.

T1 Monthly product

This product consists of a cycle forward event for $5,000 a month for the T1 service.

T1 Cancellation product

This product consists of a one time cancellation fee of $5,000, triggered when the product is canceled. It is set up with a valid date range of 365 days. Thus, 365 days after the plan is purchased, the cancellation product and the associated cancellation fee no longer apply.

Questions about the Plan

Why is the cancellation fee a subscription instead of an item?

The cancellation event is generated only when the containing product is part of a subscription product. This is different from a purchase fee which is an item.

Where do you specify that the cancellation fee is valid for one year?

In the cancel rate tier, the rate tier duration is set as a Relative Date Range. The cancel rate is valid immediately and expires in 365 days.

This means that no matter when the T1 Access Deal is purchased, the cancellation fee will apply for 365 days after the purchase date.

Web Hosting Plan

This plan (**WebHosting.IPL)** charges customers for Web Hosting space. The customer is given 100 MB of Web Hosting space for $40 a month. If a customer is using more than 100 MB at the end of the month, they are sent a warning email. Each month after the first month that they exceed the limit, they are also charged 10 cents for every megabyte over 100 that have at the end of the month.

Concepts Illustrated

No usage charge up to a threshold, warning email, delayed charge for excess usage, bill excess over threshold.

Customization Required for the Web Hosting Plan

The Web Hosting plan includes the custom service **/service/diskspace/**, a custom event called Used More Disk Space, and two different custom resources:

- Warning (resource ID = 1000011)
- Peak MBs (resource ID = 1000503)

You must add this custom service, this custom event, and these custom resources to your BRM database for the plan to work.

Adding custom services

To add the **/service/diskspace/** custom service to your BRM database, use the Developer Workshop.

Adding custom events

To add the custom event Used More Disk Space, you must configure the DM, set the proper flag in Developer Workshop, map the event to the **/service/diskspace** service.

Note that the Used More Disk Space event must include custom code to set the Peak MBs resource correctly.

Adding custom resources

To add the custom resources:

1. Start the Resource Editor and connect to your BRM database.
2. Ensure that your database does not include custom resources with the resource IDs 1000011 or 1000503.

   If your database does include custom resources with those resource IDs, you must create the custom resources for the Web Hosting plan with different resource IDs and then change the plan to refer to those new resource IDs.
3. Add the resources.

Structure of the Web Hosting Plan

The plan contains one product, Web Host Product, with a rate plan, Monthly Cycle Forward Event, and a rate plan selector, Cycle Fold Event.

Monthly Cycle Forward Event rate plan

This rate plan consists of one tier, the Web Host Rate Tier, that charges $40.00 per month of Web hosting service.

Cycle Fold Event rate plan selector

This rate plan selector includes two rate tiers, a Warning Tier and a Charge For Extra MB tier.

The Warning Tier has two discount brackets. The first charges zero US Dollars if the Peak Megabytes resource balance is between 0 and 101 at the end of the month. The second increases the resource balance of the Warning resource by 1.00 if the Peak Megabytes resource balance is 101 or higher at the end of the month. This sends a warning message to the user that they have exceeded the 100 megabytes of storage offered by the plan.

The Charge for Extra MB Tier also has two discount brackets. The first charges zero US Dollars if the Peak Megabytes resource balance is between 0 and 101 at the end of the month.

The second discount bracket charges 10 cents for each megabyte used if the peak MBs used in a month is over 100. The discount bracket also credits the account $10, which is the cost of the first 100 megabytes at 10 cents a megabyte. This credit is issued so that you charge only fort he megabytes over 100.

The credit limit of the Warning Tier is set at 1.00 in the Web Host Plan. So the Charge for Extra MB Tier does not take effect until that credit limit is reached. This means the first month customers exceed the 100 MB threshold, they receive a warning email but are not charged for the excess usage.

Questions about the Plan

Why is there a quantity bracket tier in each of the Cycle Fold Event tiers that charges $0?

This is for readability. It is easier to see that there are no extra charges for the first 100 megabytes. You could remove the first tier and simply start at 101 megabytes of usage, but in an environment where price plans are often modified and used by multiple people, it is important to make the structure of the plan as clear as possible.

Why does the second discount bracket issue a credit of $10? Would it not be easier to code the Used More Disk Space custom event to track the number of megabytes over 100 used in a month?

This design enables you to quickly and easily change the number of megabytes allowed in a month. If you code the Used More Disk Space custom event to track the number of megabytes over 100 used in a month, you must update the code to change the number of megabytes allowed in a month rather than simply changing the pricing plan.

13

Setting Up Real-Time and Pipeline Pricing and Rating

This chapter describes how to configure pricing data for the real-time rating and Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Topics in this document:

About Configuring Price List Data

Before creating your price list, you must do the following tasks to configure price list data:

- Create resources. See "Setting Up Resources or Balance Elements".
- Set up offer profiles. See "Setting Up Offer Profiles".
- Create ratable usage metrics (RUMs). See "Setting Up Ratable Usage Metrics (RUMs)".
- Map event types to RUMs. See "Mapping Event Types to RUMs".
- Specify which events you must rate. See "Mapping Events and Services".
- Map the external data to the internal usage class. See "Mapping Usage Classes".
- Define and map the usage types. See "Mapping Usage Types".

Setting Up Resources or Balance Elements

Real-time rate plans and pipeline rate plans each use a different set of resources. See the following topics:

- Setting Up Resources for Real-Time Rating
- Setting Up Pipeline Manager Resources or Balance Elements

> **Note:**
>
> You must use the same resource names for real-time and batch pipeline resources.

Setting Up Resources for Real-Time Rating

If you are using Pricing Center, you must use the Resource Editor in Pricing Center to create resources before you create a price list. You can create currency and noncurrency resources.

When you create resources, you define the following:

- The resource name, such as US Dollars.
- The resource ID, such as 840. Currency resource IDs are defined in an ISO standard.
- The rounding value. For example, to round US dollars to cents, specify a rounding value of two places after the decimal point.

> **Note:**
>
> Resource Editor sets rounding only for A/R actions such as adjustments and refunds. To set rounding for rating, discounting, and taxation as well as A/R, configure rounding in the **pin_beid** file. See "Configuring Resource Rounding".

- How to round additional numbers beyond the rounding value:
  - **Up**: For example, 10.151 rounds to 10.16.
  - **Down**: For example, 10.159 rounds to 10.15.
  - **Nearest**: If the additional digit is 0-4, the last significant digit remains the same. If the additional digit is 5-9, the last significant digit is rounded up. For example, 10.144 rounds to 10.14 and 10.145 rounds to 10.15.
  - **Even**: If the additional digit is 0-4, the last significant digit remains the same. If the additional digit is 6-9, the last significant digit is rounded up. If the additional digit is 5, the last significant digit is rounded to the nearest even digit. For example, if rounding is set to two significant digits, 10.155 rounds to 10.16 and 10.165 rounds to 10.16.
- For currency resources, you also define the error tolerance and the abbreviations and symbols used for display, such as $.
- The order in which to consume resource sub-balances. For example, if a customer's balance contains free minutes with different validity periods, you specify which minutes to use first, according to the starting validity date or ending validity date.

Setting Up Pipeline Manager Resources or Balance Elements

If you are using Pricing Center or Oracle Communications Pricing Design Center (PDC), Pipeline Manager resources or balance element IDs must match other BRM resources or balance element IDs. For example, when defining pipeline charges, you use BRM resources or balance element IDs, such as US Dollars (840). Pipeline Manager resources or balance elements can be mapped to other BRM resources or balance elements.

Defining Currencies

You must define all valid currencies before setting up currency balance elements.

You define Pipeline Manager currencies in Pricing Center, Oracle Communications Pipeline Configuration Center (PCC), or PDC.

You must also configure the DAT_Currency module. This module converts currency symbols to numeric values and retrieves balance element rounding rules, using data from **/config/beid** objects in the BRM database.

Defining a Resource or Balance Element

Resources in Pricing Center or PCC and Balance Elements in PDC are arbitrary measurement values for which a separate charge can be calculated. You can define two types of balance elements: **Money** and **Other**. **Money** balance elements are based on a defined currency. **Other** balance elements are noncurrency balance elements, such as duration or loyalty points.

You define Pipeline Manager balance elements in Pricing Center, PCC, or PDC.

Defining Currency Exchange Rates

You must set up an exchange rate for each currency you support for rating. Exchange rates specify the rate for converting one currency into another.

Call details records (CDRs) can come from multiple networks in different countries, which use different currencies. When you use exchange rates, you can store up to three charge packets with different currencies in a single EDR. The currency types are:

- **Rating currency**: The currency defined in the charge and pricing.
- **Billing currency**: The currency associated with a specific customer, network operator, and so forth for billing and invoicing.
- **Home currency**: The Pipeline Manager system-wide currency.

You convert currencies from the rating currency to the billing currency, the home currency, or both.

You define exchange rates in Pricing Center or PDC. Each exchange rate includes the following data:

- The currency to convert from.
- The currency to convert to.
- The conversion rate.
- The date from which the conversion rate is valid.

To implement currency conversion, you configure the FCT_ExchangeRate module and the DAT_ExchangeRate module.

When you configure FCT_ExchangeRate, you specify the exchange rate conversion mode by setting the **Mode** entry to Normal or Reverse. In Normal mode, the exchange rate is the multiplier to convert the From Currency to the To Currency. In Reverse mode, the exchange rate is the multiplier to convert the From Currency to the To Currency and the divisor to convert the To Currency back to the From Currency.

For example, suppose the exchange rate to convert from EUR to USD is defined as 2.0. One pipeline converts EUR to USD with **Mode** set to Normal and the another pipeline converts USD to EUR with Mode set to Reverse. In Normal mode, the exchange rate 2.0 is the multiplier to convert from EUR to USD (EUR * 2.0 = USD). In Reverse mode, the exchange rate 2.0 is the divisor to convert from USD to EUR (USD / 2.0 = EUR). If Reverse mode was not used, you would define another exchange rate 0.50 to convert from USD to EUR.

To update conversion rates, you use the DAT_ExchangeRate module **Reload** semaphore. The new exchange rates overwrite the old rates.

About currency exchange validity dates

You can use the FCT_ExchangeRate module to configure how to define the validity date for currency conversion for two cases.

- Use the **RatingDateBilling** registry entry to convert from the rating currency to the billing currency.
- Use the **RatingDateHome** registry entry to convert from the rating currency to the home currency.

You can specify a different value for the conversion to the billing currency and the home currency. For example, the validity date for converting the rating currency to the billing currency is usually the system time. The validity date for converting the rating date to the home currency is usually the CDR date.

For both registry entries, the values are:

- SYSTEM. The system time.
- FILE. The time that the file that includes the CDR was created (the file header time stamp).
- CDR. The time that the CDR was generated.
- NONE. (default).

> **Note:**
>
> If you specify **RatingDateBilling** in addition to the **RatingDateHome** and **HomeCurrency** parameters, it converts the currency to the home currency only.

Mapping Pipeline Manager Currencies and Real-Time Rating Currencies

Currencies are stored as strings for Pipeline Manager and as numbers for real-time rating. You must map Pipeline Manager currency IDs to real-time rating currency names. To do so, you create a list of currencies in the FCT_BillingRecord module registry.

Setting Up Offer Profiles

If you are using Pricing Center, set up each offer profile in the following way:

> **Note:**
>
> If you are using PDC, see the discussion about configuring policy specifications in *PDC Creating Product Offerings*.

1. Provide a unique name for the offer profile.

   > **Note:**
   >
   > When setting up a product or a discount for an existing offer profile, use the offer profile name as the provision tag of the product or discount.
   >
   > Alternately, when you create an offer profile for an existing product or discount, use the provisioning tag of the product or discount as the name for the offer profile.

2. Configure a set of policy labels that define the currency or noncurrency resource and the quality of service in rate tiers.

   For example, you can define a policy label called *Fair Usage* for a noncurrency resource (such as *Megabytes Used*, whose resource Id is 100009). Set profile status levels based on predefined quality of service. The gradation may be as follows:

   - *Low QoS* for usage between 0 and 2.5 Megabytes
   - *Medium QoS* for usage between 2.5 and 5 Megabytes
   - *High QoS* for usage above 5 Megabytes

   Example 13-1 shows the contents of one sample policy label.

Example 13-1 Sample Policy Label

```
<policy_label>
   <label>Fair Usage</label>
   <resource_name>Megabytes Used</resource_name>
   <resource_id>100012</resource_id>
   <unit> 1 </unit>
   <tier>
       <tier_start_range>0</tier_start_range>
       <tier_end_range>2.5</tier_end_range>
       <status_label>High Qos</status_label>
       </tier>
    <tier>
       <tier_start_range>2.5</tier_start_range>
       <tier_end_range>5</tier_end_range>
       <status_label>Med Qos</status_label>
    </tier>
    <tier>
       <tier_start_range>5</tier_start_range>
       <status_label>Low Qos</status_label>
```

```
    </tier>
</policy_label>
```

Setting Up Ratable Usage Metrics (RUMs)

For background information about RUMs, see "About Ratable Usage Metrics".

Real-time rate plans and pipeline rate plans each use a different set of RUMs. See the following topics:

- About Setting Up Rums for Real-Time Rating
- Setting Up Pipeline Ratable Usage Metric (RUM) Groups

About Setting Up Rums for Real-Time Rating

If you are using Pricing Center, before you create your price list, you must define the RUMs available for rating. When you define RUMs, you define the following RUM attributes:

- The event type that can be rated by using the RUM (for example, Internet usage events). You can specify more than one RUM for an event type.
- The unit of measurement. For example, to rate duration, you might specify seconds or minutes. To measure bytes, you might specify megabytes or kilobytes.
- The RUM name (for example, “Duration," “Pages," or “Bytes"). The RUM name is displayed in Pricing Center.
- How to calculate the quantity. For example, to calculate the length of a session event, you subtract the time the event started from the time that the event ended:

  Event start: 7:00 p.m.

  Event end: 9:00 p.m.

  Event end - Event start = 2 hours

  This RUM is defined as follows:

  ```
  /event/session : Duration : PIN_FLD_END_T-PIN_FLD_START_T : second
  ```

  – The RUM name “Duration" appears in Pricing Center.
  – The duration is calculated by subtracting the start time from the end time.
  – The duration is calculated by using seconds.

To perform calculations, you can use simple arithmetic by using the following operators:

- +
- -
- *
- /
- (
-)

You can use three data types:

- PIN_FLDT_INT
- PIN_FLDT_DECIMAL

- PIN_FLDT_TSTAMP

You can use fields in a substruct, but you cannot use arrays.

> **Note:**
>
> The data used for the quantity calculation is stored in event object fields. For example, the event start and end times are stored in PIN_FLD_START_T and PIN_FLD_END_T. You must be familiar with events and event fields to specify quantity calculations.

To define the RUMs, you edit the **pin_rum** file and load it into the BRM database using the **load_pin_rum** utility. See "Creating Ratable Usage Metrics".

Creating Ratable Usage Metrics

To create RUMs, you edit the **pin_rum** file and then run the **load_pin_rum** utility to load the contents of the file into the **/config/rum** object in the BRM database.

> **Note:**
>
> The **load_pin_rum** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

To create ratable usage metrics:

1. Edit the **pin_rum** file in *BRM_Home***/sys/data/pricing/example**. The **pin_rum** file includes examples and instructions.

   > **Note:**
   >
   > A RUM name can include a maximum of 255 characters.

   > **Note:**
   >
   > The **load_pin_rum** utility overwrites existing RUMs. If you are updating RUMs, you cannot load new RUMs only. You must load complete sets of RUMs each time you run the **load_pin_rum** utility.

2. Save and close the **pin_rum** file.
3. Use the following command to run the **load_pin_rum** utility:

   ```
   load_pin_rum pin_rum_file
   ```

   If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

   ```
   load_pin_rum BRM_Home/sys/data/pricing/example/pin_rum_file
   ```

For more information, see "load_pin_rum".

4. Stop and restart the Connection Manager (CM). If necessary, stop and restart Pricing Center.

To verify that the **pin_rum** file was loaded, you can display the **/config/rum** object by using Object Browser, or use the **robj** command with the **testnap** utility.

> **Note:**
>
> Fold events cannot use custom RUMs; therefore, do not assign custom RUMs to fold events in any rate plans. Products configured with custom fold RUMs are rated incorrectly.

Setting Up Pipeline Ratable Usage Metric (RUM) Groups

RUMs are ways in which you can measure events, such as how long they last or how big they are. When you create RUMs, you give them names, such as Duration or Volume, and assign them a type, such as Duration, Event, or Normal.

Units of measurement (UoMs) are the names you give to units you will use to measure events, such as seconds or bytes.

A RUM group associates two or more RUMs together and associates a UoM with each RUM in the group.

When you rate events by using a RUM group, a separate charge packet is created for each RUM.

About Defining Units of Measurement (UoMs)

UoMs are the names you give to units you will use to measure events, such as seconds or bytes.

About Defining Ratable Usage Metrics (RUMs)

RUMs are types of event measurements that you define, such as duration or volume.

About Defining a RUM Group

A RUM group groups two or more RUMs together and associates a UoM with each RUM in the group.

Converting Units of Measurement

The UoM of an incoming EDR can be converted into a UoM needed for rating a particular service. For example, an EDR might include the usage amount in seconds, but the service is configured to charge by minutes. The FCT_UoM_Map module converts seconds to minutes, using rounding rules.

To configure UoM mapping:

1. Use Pricing Center or PCC to create a UoM map to convert UoMs. When you convert UoMs, you specify the following:

- The RUM that uses the UoM.
- The UoM to convert from.
- The UoM to convert to.
- The conversion factor. For example, use a factor of 1024 to convert kilobytes into bytes.
- The rounding rule. The options are:
  - Nearest (N)
  - Always up (U)
  - Always down (D)

2. Configure the FCT_UoM_Map module.

Mapping Event Types to RUMs

If you are using Pricing Center for creating product offerings, you map event types to RUMs by using Pricing Center. However, you can edit the **pin_usage_map** file and load its content into the BRM database by running the **load_usage_map** utility. By doing so, you update the **/config/usage_map/system** object.

> **Note:**
>
> If you are using PDC, see the discussion about setting up the service-event map in *PDC Creating Product Offerings*.

> **Note:**
>
> The **load_usage_map** utility requires a configuration (**pin.conf**) file in the directory from which you run the utility.

1. Edit the **pin_usage_map** file in *BRM_Home***/sys/data/pricing/example**. The **pin_usage_map** file includes examples and instructions.

   > **Note:**
   >
   > The **load_usage_map** utility overwrites existing usage maps. If you are updating a set of usage maps, you cannot load new usage maps only. You must load complete sets of usage maps each time you run the **load_usage_map** utility.

2. Save the **pin_usage_map** file.
3. Use the following command to run the **load_usage_map** utility:

   ```
   load_usage_map pin_usage_map_file
   ```

   If you are not in the same directory as the **pin_usage_map** file, include the complete path to the file. For example:

```
load_usage_map BRM_Home/sys/data/pricing/example/pin_usage_map_file
```

For more information, see "load_usage_map".

4. Stop and restart the Connection Manager (CM). If necessary, stop and restart Pricing Center.

To verify that the data loaded correctly, display the **/config/usage_map/system** object by using Object Browser, or use the **robj** command with the **testnap** utility.

About Mapping Services

Incoming event data records (EDRs) from multiple switches often use different codes to represent the same service or supplementary service. To process EDRs, you must normalize that data by mapping external service codes to internal service codes, service classes, and usage classes. You also must map usage types so they can be added to EDRs.

- An internal *service code* represents a service such as voice or fax.
- An internal *service class* represents a variation on the service (for example, a GPRS quality of service (QoS) or a telephone service used only for emergency calls).
- An internal *usage class* represents a supplementary service (for example, call forwarding or a voice mail box). The data used for rating usage classes comes from the network.
- A *usage type* represents an attribute of a customer's account, such as an extended rating attributes (ERA), that is used in rating but which is not available from the network.

You define the internal codes and usage types in PCC. All of these are used to rate usage.

> **Note:**
>
> If you are using PDC, see the discussion about setting up service and event definitions in *PDC Creating Product Offerings*.

To create internal service mappings, you use PCC to set up the data that specifies how to map the external data to the internal codes. For example, you can specify that if the external service code is **011**, the internal service code is **TEL**.

The mapping data you enter in PCC must match the syntax and format of the EDR input data.

About Creating Map Groups

You organize service codes, service classes, and usage classes in *map groups*. A map group specifies the set of services and usage scenarios to be used for rating by Pipeline Manager. You specify the name of the map group when setting up the pipeline modules.

You can create any number of map groups. When you have multiple map groups, you can use a different set of service mappings with different pipelines. For example:

- You rate services that use various EDR formats by setting up a pipeline for each format.
- If you rate roaming usage, you might set up three pipelines using these service mapping scenarios:
  - A retail rating pipeline that uses service mappings and usage scenarios relevant to your customer's accounts and services, such as friends-and-family calls, birthday calls, messaging services, and call forwarding.

- Two pipelines (outcollect and incollect) for rating roaming usage that includes the services and usage scenarios described in your roaming agreements.

When you configure modules that use the map group, you enter the map group name in the registry. For example:

```
ServiceCodeMapping
{
  ModuleName = FCT_ServiceCodeMap
  Module
  {
    Active = True
    DataConnection = integrate.DataPool.Database
    MapGroup = serviceMapGroup
  }
}
```

> **Note:**
>
> The map group name that you enter in the registry must exist in a map group configuration in the Pipeline Manager database.

Mapping Service Codes and Service Classes

To create service code and service class mappings, you use Pricing Center or PCC to specify which external data is used to determine the internal service codes and service classes. For example, you can specify that if the external service code is **011**, the internal service code is **TEL**.

The mappings are based on the following data:

- External service code
- Usage class
- Location area indicator
- VAS event charge offer code
- Quality of service requested
- Quality of service used
- Record type

The mappings are stored in the IFW_SERVICE_MAP table. The mapping is performed by the FCT_ServiceCodeMap module.

In addition to mapping external service codes to internal service codes, you also map internal Pipeline Manager service codes to service types defined in the BRM database. These mappings are stored in the IFW_SERVICE database table.

The configuration in Table 13-1 shows the mapping for three services:

Table 13-1 Service Mapping Configuration

Pipeline Manager Service Code	BRM Service Type
TEL	/service/telco/gsm/telephony

Table 13-1 (Cont.) Service Mapping Configuration

Pipeline Manager Service Code	BRM Service Type
SMS	/service/telco/gsm/sms
GPRS	/service/ip/gprs

Multiple Pipeline Manager service codes can be mapped to a single BRM service type. For the best performance, use as few service mappings as possible. For example, map the TEL, SMS, DATA, and FAX Pipeline Manager service codes to a single **/service/telco/gsm** BRM service type.

About Setting Up Service Mapping

To set up service mapping, do the following:

1. Create internal service codes and service classes in Pricing Center or PCC.
2. Map the external data to the internal service codes and service classes in Pricing Center or PCC.
3. Configure the FCT_ServiceCodeMap module.

   This module maps call data to service codes by using the data you entered in Pricing Center or PCC.

Mapping Events and Services

When you create your product offerings, the usage events that you specify must be associated with services in the Pipeline Manager IFW_REF_MAP table. This table specifies the service type to event type mapping so that services are associated with usage events. In addition, the IFW_REF_MAP database table also specifies the prefix for the data that identifies an account (for example, MSISDN or IMSI). See "Configuring Account ID Prefixes".

You can map multiple Pipeline Manager service codes to a single BRM service type. For example, you could map the TEL, SMS, DATA, and FAX Pipeline Manager service codes to a single **/service/telco/gsm** BRM service type.

The data is stored in the Pipeline Manager database.

Mapping an Event Type to a BRM Service Type

You can map one event type to a single BRM service type (that is, one event type to a Pipeline Manager service code). For example, to rate GPRS and GSM usage events, you map the services as shown in Table 13-2:

Table 13-2 Mapping an Event Type to a Service Type

Service	Event
/service/telco/gsm/telephony	/event/delayed/session/gprs
/service/ip/gprs	/event/delayed/session/telco/gsm

You enter service-to-event mappings in Pricing Center or PCC.

Mapping Multiple Event Types to a BRM Service Type

To rate multiple events for a service, you map multiple event types to a single service type. For example, to rate GSM and GSMTEL usage events for GSM Telephony service, you map the **/event/delayed/session/telco/gsm** and **/event/delayed/session/telco/gsmtel event** types to the **/service/telco/gsm/telephony** BRM service type.

Rating multiple events for a service involves the following:

1. Enabling the BRM Database to Store Multiple Event Types per Service
2. Populating the Event Types in the EDR Container
3. Configuring Output Streams Based on Service Code and Event Types for iRules

Enabling the BRM Database to Store Multiple Event Types per Service

To map multiple event types to a service type, you must update the one-to-one event type and service type mapping restriction in the BRM database. To do so, update the IFW_REF_MAP table.

By default, the IFW_REF_MAP table holds a PK_IFW_REM primary key constraint on the ID and the REF_OBJ columns. To update the one-to-one event type and service type mapping restriction in the BRM database, hold (or place) the PK_IFW_REM primary key constraint on the ID, REF_OBJ, and REF_PARAM columns.

Populating the Event Types in the EDR Container

When multiple event types are mapped to a service type, you must ensure that each event type is mapped to the appropriate service code.

You create an iScript or an iRule that will determine the correct event type for a service type in an EDR and populate the DETAIL.EVENT_TYPE field with the event type *before* passing the EDR through the following modules:

- FCT_Account module.
- FCT_AccountRouter module for the account router instance of the Pipeline Manager.

Configuring Output Streams Based on Service Code and Event Types for iRules

By default, the IRL_EventTypeSplitting iRule sends EDRs to separate output streams based on the service codes. When you configure multiple event types for a service type, you also map a service code to multiple event types. Therefore, you must update the IRL_EventTypeSplitting iRule and the data file to use the event types that are populated in the DETAIL.EVENT_TYPE field along with the service code to send EDRs to separate output streams.

To configure the output streams based on service codes and event types:

1. Update the IRL_EventTypeSplitting iRule to add a condition for checking the event types.

   The following IRL_EventTypeSplitting iRule sample shows the additional condition to check the event types:

   ```
   CONDITION:
   edrString(DETAIL.INTERN_SERVICE_CODE) =~ "${1}";
   edrString(DETAIL.EVENT_TYPE) =~ "${2}";
   ```

2. Edit the **IRL_EventTypeSplitting.data** file to include the event types. For example, the following entries map the GSM service code to the TELOutput and SMSOutput streams:

```
GSM;/event/delayed/telco/gsm/telephony;TELOutput
GSM;/event/delayed/telco/gsm/sms;SMSOutput
```

Example Procedure to Map Multiple Event Types to a Service Type

The following procedure is an example of how you map the **/event/delayed/session/telco/gsm/telephony** and **/event/delayed/session/telco/gsm/sms** event types to a single **/service/telco/gsm** service type.

1. Update the IFW_REF_MAP database table to hold (or place) the PK_IFW_REM primary key constraint on the ID, REF_OBJ, and REF_PARAM columns.
2. This step is optional. In Pricing Center or PCC, map the **/service/telco/gsm** service type to two internal service codes, GSM1 and GSM2.
3. Map the **/event/delayed/session/telco/gsm/telephony** and the **/event/delayed/session/telco/gsm/sms** event types to the **/service/telco/gsm** service type.
4. Update **/config/event_map** to map the **/event/delayed/session/telco/gsm/telephony** and the **/event/delayed/session/telco/gsm/sms** event types to the **/service/telco/gsm** service type.
5. Create a charge offer with a **/service/telco/gsm** service type and configure the following two event types:
   - **/event/delayed/session/telco/gsm/telephony**
   - **/event/delayed/session/telco/gsm/sms**
6. Map the **/event/delayed/session/telco/gsm/telephony** and the **/event/delayed/session/telco/gsm/sms** event types to the required pipeline charge.
7. Create an iScript called **ISC_EventType.isc** to contain the logic that populates the DETAIL.EVENT_TYPE field with the appropriate event type.
8. Update the IRL_EventTypeSplitting iRule to map each combination of service code and event type to an output stream as follows:

```
GSM;/event/delayed/telco/gsm/telephony;TELOutput
GSM;/event/delayed/telco/gsm/sms;SMSOutput
GSM1;/event/delayed/telco/gsm/telephony;TEL1Output
GSM1;/event/delayed/telco/gsm/sms;SMS1Output
```

Mapping Usage Classes

To create usage class mappings, you use Pricing Center or PCC to set up the data that specifies how to map the external data to the internal usage class. For example, you can specify that if the external supplementary service code for call forwarding is **41**, the internal usage code is **CW**. In addition, you can use various EDR attributes to create variations on supplementary service for rating purposes. For example:

- **Service code mapping**. Use the usage class to create special virtual services and charge offers; for example, you can use the combination of telephony and call forwarding to create value-added services.
- **ServiceClass-Mapping**. Use the usage class to create special sub-services or quality of services (for example, service-level agreements).
- **UsageScenario-Mapping**. Use the usage class to map to impact categories.

- **RateAdjustment**. Use the usage class to provide discounts or adjustments in individual EDRs.

The mappings are based on the following data:

- External usage class
- Wholesale zone
- Tariff class and subclass
- Record type
- Connect type and Connect subtype, for example, anonymous, from another network, or direct
- Transit area code
- APN address when you use GPRS or UMTS technology
- SS (supplementary service) packet

The mapping is performed by the FCT_UsageClassMap module. You can configure this module to use a specific map group.

About Setting Up Usage Class Mapping

To set up usage class mapping:

1. Create internal usage classes in Pricing Center or PCC.
2. Map the external data to the internal usage classes in Pricing Center or PCC.
3. Configure the FCT_UsageClassMap module.

   This module maps usage classes by using the data you entered in Pricing Center or PCC.

Mapping Usage Types

A usage type represents an attribute of a customer's account that is used in rating but which is not available from the network (for example, closed usergroup calls, friends-and-family calls, VIP calls, and birthday calls). A usage type is dynamically determined by comparison patterns using the A number and the B number.

You define usage types to represent the following customer information in an EDR:

- An ERA, such as an ERA for Friends and Family calls.

  The usage type used for rating ERAs is determined by the call origin and destination data in the EDR. For example, if the destination number in an EDR matches the number on the account's friends-and-family list, the corresponding friends-and-family usage type is added to the EDR and is used during rating.

- A group to which a customer belongs.

  This usage type is used by filter sets to apply system charge offers and system discounts to a select group of customers. For example, if the EDR's DETAIL.CUST_A.CUST_SEG_LIST field matches one or more values you define, the corresponding usage type is added to the EDR.

> **Note:**
>
> You must create your own application to add data to the CUST_SEG_LIST field.

The usage types that you define are available to all rating scenarios.

Usage types are added to EDRs by running the IRL_UsageType iRule. This iRule performs no rating, it just adds the usage type to the EDR DETAIL block. You specify the rules for mapping EDR data to usage types in the files used by the IRL_UsageType iRule.

About Setting Up Usage Type Mapping

To set up usage type mapping:

1. Configure the IRL_UsageType iRule files. See the following sections:
   - Configuring the IRL_UsageType iRule for ERAs
   - Configuring the IRL_UsageType iRule for Filter Sets
   - Configuring the IRL_UsageType iRule for Both ERAs and Filter Sets
2. Configure the FCT_IRule module. See "Configuring the FCT_IRule Module to Run the IRL_UsageType iRule".
3. Create usage types in Pricing Center or PCC.

Configuring the IRL_UsageType iRule for ERAs

The IRL_UsageType iRule uses the following files in the *pipeline_home*/**iScriptLib** directory. The directory contains sample files.

- **ISC_UsageType.isc**. This file contains the iScript code that determines whether an EDR qualifies for a usage type. This script reads EDR fields and sets a series of variables. It then assigns a usage type to the EDR based on a mapping of variables to usage type.

  You can modify the **ISC_UsageType.isc** file to customize how to enter ERA data in Customer Center. For example, the following line uses a date format of *YYYY.MM.DD*:

  ```
  String strCallCombineYearMonth = "." + strCallMonth + "." + strCallDay;
  ```

  You can change this iScript to support a date format of *YYYY/MM/DD* by doing the following:

  ```
  String strCallCombineYearMonth = "/" + strCallMonth + "/" + strCallDay;
  ```

- **IRL_UsageType.irl**. This file contains the rules for mapping the variables to values. If the EDR contains an attribute that matches a variable, the value for that variable is set to **Yes**. Otherwise it is set to **No**.
- **IRL_UsageType.data**. This file contains the rules for assigning a usage type to the EDR. If the values of the variables match those specified in this file, the corresponding usage type is set in the EDR.

Configuration example

> **Note:**
>
> The sample configuration shows how to configure the iRule for ERAs only. It does not show how to configure the iRule for filter sets, which is done differently. For information, see "Configuring the IRL_UsageType iRule for Filter Sets" and "Configuring the IRL_UsageType iRule for Both ERAs and Filter Sets".

ISC_UsageType.isc

For ERAs, this file needs determination logic as shown in the following examples for the Friends and Family and Customer to Customer ERAs, respectively. This is in addition to the included **ISC_UsageType Function.isc** file, which is run when a condition is fulfilled.

```
isFriendsAndFamily = compareProductERAforProfileAttributes
(profileNameFriendsFamily,"NUMBER",
edrString(DETAIL.B_NUMBER), serviceType, EXACT_MATCH);

if  (edrNumDatablocks( DETAIL.CUST_A) > 0 and
edrNumDatablocks(DETAIL.CUST_B) > 0)
isCustomerToCustomer = "Y";
```

IRL_UsageType.irl

The following example shows the rules (conditions) for mapping variables to values for all ERAs. It also shows the result placeholder for the usage type. Each number corresponds to a position in the **IRL_UsageType.data** file.

```
CONDITION:
isRoaming =~ "${1}";
isInternational =~ "${2}";
isHomeRegion =~ "${3}";
isCustomerToCustomer =~ "${4}";
isSameClosedUserGroup =~ "${5}";
isSameCorporateAgreement =~ "${6}";
isSameCustomer =~ "${7}";
isSameSystemBrand =~ "${8}";
isSameSegment =~ "${9}";
isSameRateplan =~ "${10}";
isSameDiscountModel =~ "${11}";
isSameBillcycle =~ "${12}";
isBirthdayCall =~ "${13}";
isFriendsAndFamily =~ "${14}";
isHomeCell =~ "${15}";

RESULT:
edrString( DETAIL.USAGE_TYPE ) = "${16}";
```

IRL_UsageType.data

Each line in the **IRL_UsageType.data** file represents a different usage type mapping. Each position in a line corresponds to a variable (condition) in the **IRL_UsgeType.irl** file. The last position specifies the usage type code to apply if the mapping matches. This position corresponds to the result variable in the **IRL_UsageType.irl** file, which in the example above is “${16}";. Each position can have one of the following values:

- **N** = No
- **Y** = Yes
- **.** = any value

The following examples show the mapping and usage type for the Friends and Family and Customer to Customer ERAs, respectively.

```
N;N;.;.;.;.;.;.;.;.;.;.;Y;.;FNF

N;N;.;Y;.;.;.;.;.;.;.;.;.;.;CTC
```

Using the configuration examples above:

- If an EDR is not roaming or international (N in positions 1 and 2) and uses the Friends and Family ERA (Y in position 14), the usage type **FNF** is assigned.

  This mapping is associated with these conditions in the **IRL_UsageType.irl** file:

  ```
  isRoaming =~ "${1}";
  isInternational =~ "${2}";
  isFriendsAndFamily =~ "${14}";
  ```

- If an EDR is not roaming or international (N in positions 1 and 2) and uses the Customer to Customer ERA (Y in position 4), the usage type **CTC** is assigned.

  This mapping is associated with these conditions in the **IRL_UsageType.irl** file:

  ```
  isRoaming =~ "${1}";
  isInternational =~ "${2}";
  isCustomerToCustomer =~ "${4}";
  ```

Configuring the IRL_UsageType iRule for Filter Sets

The IRL_UsageType iRule uses the following files in the *pipeline_home*/**iScriptLib** directory. There are samples of these files in that directory.

- **ISC_UsageType.isc**. This file contains the iScript code that determines whether an EDR qualifies for a usage type. This script reads EDR fields and sets a series of variables. It then assigns a usage type to the EDR based on a mapping of variables to usage type.
- **IRL_UsageType.irl**. This file contains a mapping to the contents of the **IRL_UsageType.data** file. For filter sets, this is a condition placeholder and a result placeholder.
- **IRL_UsageType.data**. This file contains one or more conditions and the usage type to be assigned. If the EDR contains attributes that match the conditions specified in this file, the corresponding usage type is set in the EDR.

Configuration example

> **Note:**
>
> This sample configuration shows how to configure the iRule for filter sets only. It does not show how to configure the iRule for ERAs, which is done differently. For information, see "Configuring the IRL_UsageType iRule for ERAs" and "Configuring the IRL_UsageType iRule for Both ERAs and Filter Sets".

ISC_UsageType.isc

For filter sets, all you need is the included **ISC_UsageType Function.isc** file, which is run when a condition is fulfilled.

IRL_UsageType.irl

The following example shows a placeholder condition that indicates the position in the **IRL_UsageType.data** file that contains the determination logic for applying the usage type. This example also shows a result placeholder for the usage type.

> **Note:**
>
> Do not include a string in the condition to represent the EDR field being checked. For filter sets, the fields are identified in the **IRL_UsageType.data** file.

```
CONDITION:
"${1}";

RESULT:
edrString( DETAIL.USAGE_TYPE ) = "${2}";
```

IRL_UsageType.data

Each line in the **IRL_UsageType.data** file represents a different usage type mapping. Each position in a line corresponds to the positions in the **IRL_UsageType.irl** file.

The following examples show rules for assigning a usage type to the EDR. For filter sets, the rule contains both the determination logic and the usage type to apply. You can use Boolean operators to define complex rules.

```
isSegmentContains("1234") and isSegmentContains("GOOD");A1

isSegmentContains("789") or isSegmentContains("OK");B2
```

Using the configuration examples above:

- If the CUST_SET_LIST field is set to **1234** *and* **GOOD**, the usage type **A1** is assigned.
- If the CUST_SET_LIST field is set to **789** *or* **OK**, the usage type **B2** is assigned.

Configuring the IRL_UsageType iRule for Both ERAs and Filter Sets

You can configure one set of IRL_UsageType iRule files for both ERA and filter set usage type mapping. For background information, see "Configuring the IRL_UsageType iRule for ERAs" and "Configuring the IRL_UsageType iRule for Filter Sets".

ISC_UsageType.isc

There is no special configuration in this file when you use it for both ERA and filter set usage type mapping. You need both the determination logic for ERAs and the included **ISC_UsageType Function.isc** file, which is run when a condition is fulfilled.

IRL_UsageType.irl

Combine both ERA variables and a placeholder condition for filter sets (${16}; in this example) in the **IRL_UsageType.irl** file. You need only one result entry.

```
CONDITION:
isRoaming =~ "${1}";
isInternational =~ "${2}";
isHomeRegion =~ "${3}";
isCustomerToCustomer =~ "${4}";
isSameClosedUserGroup =~ "${5}";
isSameCorporateAgreement =~ "${6}";
isSameCustomer =~ "${7}";
isSameSystemBrand =~ "${8}";
isSameSegment =~ "${9}";
isSameRateplan =~ "${10}";
isSameDiscountModel =~ "${11}";
isSameBillcycle =~ "${12}";
isBirthdayCall =~ "${13}";
isFriendsAndFamily =~ "${14}";
isHomeCell =~ "${15}";
${16};

RESULT:
edrString( DETAIL.USAGE_TYPE ) = "${17}";
```

IRL_UsageType.data

In the **IRL_UsageType.data** file, provide a separate line for each rule.

The following is a line to assign a usage type for a filter set. The condition is in position 16 and the result is in position 17.

> **Note:**
>
> Be sure to account for each condition in the **IRL_UsageType.irl** file.

```
.;.;.;.;.;.;.;.;.;.;.;.;.;.;isSegmentContains("1234") and isSegmentContains("GOOD");A1
```

The following is a line to assign a usage type for the Friends and Family ERA. The filter set position, 16, is set to any value (.).

```
N;N;.;.;.;.;.;.;.;.;.;.;.;Y;.;.;FNF
```

You can also combine ERA and filter set conditions in one rule. You can use this to apply an ERA usage type only to certain customers or to assign a filter set usage type to specific customers who also have an ERA:

```
N;N;.;.;.;.;.;.;.;.;.;.;.;Y;.;isSegmentContains("1234") and isSegmentContains("GOOD");C3
```

Configuring the FCT_IRule Module to Run the IRL_UsageType iRule

The IRL_UsageType iRule is run by the FCT_IRules module. You must run this instance of the FCT_IRules module after the FCT_Account module and before the FCT_USC_Map module.

Adding Customer Balance Impact Data to EDRs

> **Note:**
>
> Before adding balance impacts to an EDR, you must configure internal service codes. See "About Mapping Services".

Before rating an EDR, you must add the customer's balance impact data to the EDR. This data includes the following:

- The POID of the service-level item that receives the balance impact.
- The start and end times of the accounting cycle that applies to the event. This data assures that the charges are applied to the correct bill item.

To find the balance impact data, you configure the following modules:

- Use the FCT_Account module to look up the balance impact data and add it to the EDR.
- Use the DAT_AccountBatch module to supply data to the FCT_Account module.

Configuring the DAT_AccountBatch Module

The DAT_AccountBatch module stores all customer data from the BRM database. The FCT_Account and FCT_AccountRouter modules use the data stored by the DAT_AccountBatch module.

> **Note:**
>
> Because FCT_AccountRouter runs in a separate instance of Pipeline Manager, you configure separate instances of the DAT_AccountBatch module for the FCT_Account and FCT_AccountRouter modules.

The DAT_AccountBatch module stores the following types of data:

- *Dynamic data*, such as login names, charge offers, accounts, and services. This data is updated by the DAT_Listener module.
- *Maintenance data*, such as charge offers, packages, and bundles. Maintenance data is not updated constantly. Instead, it is updated only when all data is reloaded into the database.

To configure the DAT_AccountBatch module, see the following sections:

- Specifying Not to Load Closed Accounts
- Specifying Whether to Load All Account Data
- Specifying How Much Account Data to Retrieve on Startup
- Configuring Charge Offer Validity Checking
- Configuring Account Charge Offer Validity Checking for Backdated Events

Specifying Not to Load Closed Accounts

When you start Pipeline Manager, DAT_AccountBatch loads all accounts into memory. This can affect start-up performance when there are a great number of accounts. To improve Pipeline Manager start-up performance, you can reduce the number of accounts loaded into memory by specifying not to load closed accounts.

To not load closed accounts, you set the **ClosedAccountDelay** registry entry in the DAT_AccountBatch registry. Specify the number of days before the current date for which closed accounts are not loaded:

```
ClosedAccountDelay = number_of_days
```

For example, if the system time is 11:00 a.m. on June 25 and the number of days is 10, accounts that were closed before 11:00 a.m. on June 15 are not loaded into memory.

The **ClosedAccountDelay** registry entry applies only to closed accounts. Accounts that are inactive, and open accounts with inactive services, are still loaded into memory.

If delayed EDRs arrive in the pipeline for closed accounts that were not loaded into memory, those EDRs are rejected and not rated.

The **ClosedAccountDelay** registry entry uses only the system time, not the virtual time. If you set the virtual time and restart the pipeline, the virtual time is not considered when loading closed accounts.

For example, on June 20 you set the virtual time to June 5 and create an account. You then change the virtual time to June 9, close the account, and set **ClosedAccountDelay** to 10 days. When you restart the pipeline, the account is not loaded because it was closed more than 10 days before the system time (June 20), even though the current virtual time is June 9. EDRs that arrive for that account are rejected.

Specifying Whether to Load All Account Data

Use the DAT_AccountBatch module **InitialLoading** registry entry to specify whether the initial loading of service and account data is performed. If the data is not loaded, loading occurs while processing. Login objects are always loaded.

The default is **True**. Setting this entry to **False** enables the system to start faster, but processing might be slower.

Specifying How Much Account Data to Retrieve on Startup

Use the DAT_AccountBatch module **RowFetchSize** registry entry to specify the number of rows of data to retrieve from the BRM database. The default is 1000.

The value you enter is used only during the initialization of DAT_AccountBatch to improve performance. After the DAT_AccountBatch startup is completed, **RowFetchSize** automatically resets itself to a more appropriate value of 50 to conserve memory.

Configuring Charge Offer Validity Checking

You can use the DAT_Account module **UseProductCreatedTime** registry entry to configure how charge offer validity is checked:

- If you enable this entry, the charge offer created time is not considered when establishing charge offer validity. Only the start and end times are considered, so updates to charge offers are valid immediately.
- If you disable this entry, the module checks the created time and the start and end time of a charge offer when establishing the charge offer's validity. In some situations this procedure can lead to unexpected results. The created time (PIN_FLD_CREATED_T) is system-generated whenever the charge offer is modified. If the charge offer is modified after its start time (PIN_FLD_START_T), there can be a short period when the charge offer is invalid because its created time is later than the start time.

Configuring Account Charge Offer Validity Checking for Backdated Events

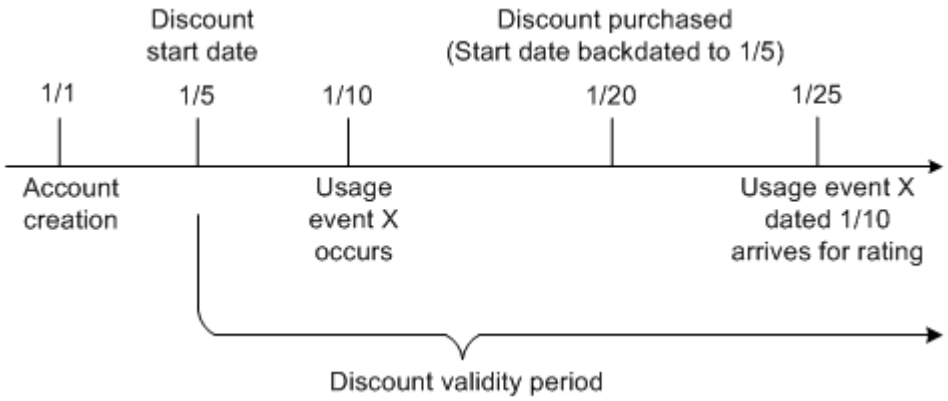
You can use the DAT_Account module **UseLatestProductAndDiscount** registry entry to configure validity checking for account charge offers and discount offers.

- If you enable this entry, when rating EDRs, DAT_AccountBatch retrieves account charge offer and discount offer information based only on their start and end dates.
- If you disable this entry, DAT_AccountBatch retrieves account charge offer and discount offer information based on their creation dates.

When charge offer and discount offer information is retrieved during EDR rating, pipeline modules validate the account charge offers and discount offers by using the account creation date. This date is the time that the charge offer and discount offer are purchased (the EFFECTIVE_T time in the AU_ACCOUNT_T audit table). If the charge offer's validity start date is earlier than the actual purchase date, which can occur when purchase events are delayed, usage events might be incorrectly rated.

For example, an account is created on January 1. On January 20, the customer purchases a discount offer with a validity period that starts on January 5. On January 25, a delayed event that is dated January 10 arrives for rating. The discount module does not apply the discount because it uses the purchase date of January 20 instead of the discount's start date of January 5. Only events dated on or after January 20 are discounted. Figure 13-1 shows a timeline for this example.

Figure 13-1 Charge Offer Validity for Backdated Events Example



If you set the **UseLatestProductAndDiscount** entry in the DAT_AccountBatch registry, account charge offer and discount audit information is not retrieved. If you want delayed events to use the account charge offer and discount offer that was valid at the time the usage occurred, do not set this entry.

Getting Information about Loading Accounts

You can use DAT_AccountBatch semaphore commands to get data about the accounts loaded into the DAT_AccountBatch module.

- **PrintData** - Reports the account data for all accounts.

  The data is written to the specified file. If you do not specify a file, it is written to **stdout**.

- **PrintDataLogin** - Reports the account data for a single account identified by the BRM login ID (usually the phone number).

  The data is written to a file named *LoginID*.**lst**.

- **PrintDataSamples** - Reports the account data for a specified number of accounts, chosen randomly.

  The data is written to a file named *Value*.**lst**. For example, if you enter **15**, the file is named **15.lst**.

  If no value is provided, the module prints data for 10 accounts to **stdout**.

Configuring the FCT_Account Module

To find the balance impact data:

1. The FCT_Account module looks for the following data in the EDR:
   - The internal service code that indicates which data can be used to identify the account that generated the EDR. For example, if the internal service code is a telephony service, the identifying data is the A number. A different service might use the IMSI as the identifier.

     You identify which data to use by using Pricing Center or PCC. See "Specifying Which Data Is Used for Identifying Accounts". Depending on the service, the module searches for either the login from a service object or an alias from a service object. An alias is typically a phone number or IMSI.

     In rare cases, one customer's MSISDN can be the same as another customer's IMSI. You can configure account ID prefixes to use for handling duplicate telephone numbers. See "Configuring Account ID Prefixes".

   - The timestamp for the EDR. The timestamp is important because telephone numbers can be used by different accounts at different times.
2. The FCT_Account module uses the DAT_AccountBatch module to look up the account.

   Note:

   - If the A customer is not found, the EDR is rejected. If the B customer is missing, no error is generated.
   - Because phone numbers can be recycled, the search is made on data from BRM audit objects.
   - Accounts are loaded based on the service that is being rated by the pipeline. If an account does not own the service that the pipeline rates, it is not loaded.
3. The DAT_AccountBatch module returns the balance impact data.

4. The FCT_Account module inserts the customer data into the EDR.

Specifying Which Data Is Used for Identifying Accounts

Different services require different ways to look up accounts. For example, to look up a telephone account, you can use the A number to find the originating account and the B number to find the account that was called.

To specify which data to use for looking up an account, you use Pricing Center or PCC to map the service to the type of data. The data is stored in the IFW_ALIAS_MAP table.

You can set up multiple mappings to support different services (for example, specify different mappings for each pipeline).

> **Note:**
>
> This data is used by both the FCT_Account module and the FCT_AccountRouter module.

Use the following data to specify how to identify accounts:

- The EDR container description that includes the field to use for identifying the account.
- The account reference. You must use one of the following:
  - Use **Account_CustA** for the A number.
  - Use **Account_CustB** for the B number.
- The internal service code (for example, **TEL** or **DATA**).
- The type (for example **Internal** or **Plugin**).

Configuring Account ID Prefixes

In some cases, the data used for identifying an account is not unique; for example, the MSISDN of one customer might have the same value as the IMSI of another customer. To avoid any conflicts, you can specify a prefix for the identifying data. This prefix data is stored in the IFW_REF_MAP database table.

You can specify a prefix based on the service type and event type, as shown in Table 13-3. In this case, the number is prefixed with either **msisdn** or **imsi**.

Table 13-3 Prefix Configurations

Service	Event	Prefix	Example of Result
/service/telco/gsm/telephony	**/event/delayed/session/gprs**	**msisdn**	**msisdn00491721234567**
/service/ip/gprs	**/event/delayed/session/telco/gsm**	**imsi**	**imsi00491721234567**

To set up prefixes:

1. Customize the PCM_OP_NUM_POL_DEVICE_ASSOCIATE and PCM_OP_SIM_POL_DEVICE_ASSOCIATE policy opcodes to add a prefix to the MSISDN or IMSI.

2. In the Pipeline Manager IFW_REF_MAP object, use the IFW_REF_MAP.REF_COL attribute to specify the prefix, for example, **msisdn**.

Specifying Which Noncurrency Sub-Balances to Load on Startup

The DAT_BalanceBatch module uses the noncurrency balance element validity to select the noncurrency sub-balances to load from the BRM database into pipeline memory. For example, if the Free Minutes balance element validity is 30 days, at Pipeline Manager startup, DAT_BalanceBatch selects all Free Minutes sub-balances that were valid for the last 30 days.

You configure the noncurrency balance element validity in the balance element configuration file (**pin_beid**). If the noncurrency balance element validity is not configured, DAT_BalanceBatch selects the sub-balances that were valid for 366 days by default

> **Note:**
>
> When you set the noncurrency balance element validity, the sub-balances that do not fall within the validity range are not loaded into pipeline memory and therefore are not available for rating or rerating of call details records (CDRs). To make available the sub-balances that are required for rating and rerating, set the balance element validity to an appropriate setting. For instance, if the CDRs are older than 366 days, you should set the validity greater than 366 days to ensure that all the sub-balances that are required are loaded and available for rating or rerating the CDRs. See "Configuring the Noncurrency Balance Element Validity".

If your business requires, you can configure DAT_BalanceBatch to load all the noncurrency sub-balances available in the BRM database. To load all the noncurrency sub-balances, use the **SelectiveSubBalLoad** entry in DAT_BalanceBatch registry. When this entry is set to **False**, DAT_BalanceBatch loads all noncurrency sub-balances, including the sub-balances that are expired, into pipeline memory.

> **Note:**
>
> When the BRM database contains a large number of noncurrency sub-balances, loading all sub-balances leads to increased Pipeline Manager startup times.

Configuring the Noncurrency Balance Element Validity

To configure the validity of a noncurrency balance element:

1. Edit the **pin_beid** file in *BRM_home***/sys/data/pricing/example**.

   Follow the instructions in the **pin_beid** file.

> **Note:**
>
> If you are using an existing **pin_beid** file, you must manually add the **Validity_in_days** field to the syntax and set the balance element validity for the noncurrency balance element. In this example, the balance element validity for balance element **1000010** is set to **30** days.
>
> ```
> #beid r_mode round tol_min tol_max tol_pct beid_str_code symbol event stage
> con_rule_id apply_mode Name Validity_in_days
> #
> 1000010 0 1 0.000000 0.000000 0.000000 MIN Min * 0 0 1 Free Domestic
> Minutes 30
> ```

> **Note:**
>
> The **load_pin_beid** utility overwrites the existing balance elements. If you are updating balance elements, you cannot load new balance elements only. You must load complete sets of balance elements each time you run the **load_pin_beid** utility.

2. Save and close the **pin_beid** file.
3. Run the following command:

   ```
   load_pin_beid pin_beid
   ```

   If you do not run the utility from the directory in which the **pin_beid** file is located, you must include the complete path to the file. For example:

   ```
   load_pin_beid BRM_home/sys/data/pricing/example/pin_beid
   ```

   > **Tip:**
   >
   > If you copy the **pin_beid** file to the directory from which you run the **load_pin_beid** utility, you do not have to specify the path or file name. The file must be named **pin_beid**.

4. Stop and restart the Connection Manager (CM).

   To verify that the noncurrency balance elements were loaded, you can display the **/config/beid** object by using Object Browser, or use the **robj** command with the **testnap** utility.

5. Start Pipeline Manager.

   At Pipeline Manager startup, DAT_BalanceBatch uses the noncurrency balance element validity in the **/config/beid** object to select the noncurrency sub-balances to load into pipeline memory.

Modifying and Loading the EDR Container Description

By default, BRM supports the following EDR container description:

- **containerDesc.dsc:** The *rating EDR container description*. Defines the format into which CDRs are converted so that they can be processed for rating by a pipeline input module.

The EDR container description is in the *pipeline_home*/**formatDesc/Portal** directory.

If the CDR requests you process include information that has no corresponding fields in the appropriate default EDR container description, you must modify the description to accommodate the custom data. See "Modifying EDR Container Descriptions".

After editing the rating EDR container description, you must load it into the appropriate Pipeline Manager database. See "Loading EDR Container Descriptions".

Modifying EDR Container Descriptions

If the CDR requests you process include information that has no corresponding fields in the default EDR container description, you must modify the description to accommodate the custom data.

The rating EDR container description (**containerDesc.dsc**) text file is located in *pipeline_home*/**formatDesc/Portal**.

EDR Container Description Format

An EDR container description has the following format:

```
ComplexDataType
{
  String             StringVar;  //description
  Integer            IntegerVar;
  Date               DateVar;
  Decimal            DecimalVar;
  ComplexDataType_2  complexVar;
}
ComplexDataType_2
{
  String             var1;
  Decimal            var2;
}
```

The following example shows how to format an EDR container description file:

```
// Comments
DETAIL // comment
{
String A_Number; // Comment will be filled into the ifw_edrc_field.description
Date STARTDATE;
Decimal duration;
CUSTOMER CUST_A; // New complex type must be defined later on
CUSTOMER CUST_B;
}
HEADER
{
// comment
String name;
Integer count;
}
CUSTOMER
{
String account;
String account_no;
Purchased_Product PRODUCT;
}
Purchased_Product
{
```

```
String name;
Integer type;
}
```

Note the following syntax and grammar rules:

- The file is case-sensitive.
- Real data types are the following:
  - String
  - Integer
  - Date
  - Decimal
  - Block (for example, DETAIL.CUST_A)
- The parsing is done in one step. Therefore, embedded complex data types must be defined later in the file.
- The name of each complex data type must be written in one line.
- The complex data type area must begin with a left brace ({) and end with a right brace (}). Each brace must be in its own line.
- Each data type variable pair must be written in one line. The line must end with a semicolon (;).
- Comments must begin with double slashes (//) and are valid until the end of the line.
- Comments following a real data type line are copied to the IFW_EDRC_FIELD.DESCRIPTION field.

Loading EDR Container Descriptions

After editing the rating EDR container description, you must load it into the Pipeline Manager database.

You use the **containerDescLoader.pl** utility to load an EDR container description into a database.

EDR container descriptions are stored in the following database tables:

- IFW_EDRC_DESC
- IFW_EDRC_FIELD

Note the following restrictions:

- The Container Description Loader utility can process only one EDR container description file at a time.
- You cannot update tables that already include data. You can initialize empty tables or delete the data from tables before adding new data. You cannot, however, delete existing data if there are references to it from elsewhere in the database.

Creating the EDR Load Utility Command File

The Container Description Loader utility uses a command file to load data into a database.

The syntax of the command file is as follows:

```
#comments
ECHO print this on default output
TYPE=Database type
SOURCE=Database source
USER=Database user
PASS=Database user password
INIT=TRUE | FALSE
EDRC_DESC=Pipeline name as specified in the registry
EDRC_DESC_FILE=Container description file
EDRC_DESCRIPTION=Description of pipeline name
RUN=Execute this command
```

This is a sample command file with only the required entries:

```
DB=PR01|zaphod|zaphod
EDRC_DESC_FILE= /FMD/Portal/containerDesc.dsc
EDRC_DESC='MY_IFW'
EDRC_DESCRIPTION='MY_IFW container description'
```

This sample includes optional entries:

```
##
## This is a sample command file for containerDescLoader.pl
##
##
## DB (database connect)
## database|user|password
##
DB=PR01|zaphod|zaphod
#
# initialization (deleting tables)
#
ECHO=###################################
ECHO=# deleting of ifw_edrc_desc and ifw_edrc_field
ECHO=###################################
#INIT=FALSE
INIT=TRUE
# container description file
EDRC_DESC_FILE= /FMD/Portal/containerDesc.dsc
# pipeline name as specified in the registry
# filled into IFW_EDRC_DESC.EDRC_DESC and IFW_EDRC_FIELD.EDRC_DESC
EDRC_DESC='MY_IFW'
# additional description
# filled into IFW_EDRC_DESC.NAME
EDRC_DESCRIPTION='MY_IFW container description'
# optional command processing
#RUN=prog
```

Before Running the EDR Load Utility

Before running the Container Description Loader utility, do the following:

- Ensure that the following components are installed:
  - Perl 5.004 or higher
  - CPAN generic database interfaces
  - **oraperl** Perl module
- If necessary, create the following database tables:
  - IFW_EDRC_DESC

  - IFW_EDRC_FIELD
- If necessary, create the following sequence:
  - IFW_SEQ_FIELD_ID
- Use the **IFW_PERL_LIB** variable to find the required Perl modules.

Starting the EDR Load Utility

To start the Container Description Loader utility:

1. Go to the **TLS_ContainerDescLoader** directory.
2. Enter the following command:

   containerDescLoader.pl [**-c** *command_file*].

Dropping or Upgrading Incomplete Calls after Changing the EDR Container Description

The FCT_CallAssembling module stores any incomplete call records while a pipeline is running. When you stop the pipeline and change its container description, these incomplete call records are then automatically in the wrong format for the pipeline to process. This section describes your two choices for processing these call records:

- Discard them. This is the fastest way of dealing with the problem, but you lose the revenue they represent. You can however, use these EDRs for auditing purposes. For the procedure, see "Discarding Incomplete Calls after Changing the EDR Container Description".
- Upgrade them to the new container description. You use the tools provided to create a data upgrade pipeline that reformats the EDRs into the new container description format. For the procedure, see "Upgrading Incomplete Calls to the New Container Description".

These sections apply to all pipelines using FCT_CallAssembling to assemble CDRs into EDRs for the pipeline to process.

Discarding Incomplete Calls after Changing the EDR Container Description

This section explains the steps necessary to delete your partially-assembled call records during the process of changing the EDR container description.

You use the **UpgradeFlushLimit** semaphore registry entry to FCT_CallAssembling to flush and discard any incomplete calls stored in a pipeline during an EDR container description change.

You then use the **EmitPartialEDROnUpgrade** startup registry entry to specify what happens to the EDRs flushed by the **UpgradeFlushLimit** semaphore:

- If the **EmitPartialEDROnUpgrade** registry entry is **False**, the EDRs are silently dropped and are never processed by the pipeline.
- If the **EmitPartialEDROnUpgrade** registry entry is **True**, partial EDRs are processed by the pipeline but are not rated. You can use these records for revenue assurance purposes, but they should be discarded to avoid recycling.

EDR Data Available for Auditing

By default, all flushed EDRs will contain the following data fields:

- CHAIN_REFERENCE
- LONG_DURATION_INDICATOR (set to **P**)
- NUMBER_OF_CDRS
- CHARGING_START_TIMESTAMP
- CHARGING_END_TIMESTAMP
- DURATION

The following additional fields will also be included if the FCT_CallAssembling module has **AssembleVolume** set to **True**:

- VOLUME_SENT
- VOLUME_RECEIVED
- NUMBER_OF_UNITS
- RETAIL_CHARGED_AMOUNT_VALUE
- WHOLESALE_CHARGED_AMOUNT_VALUE

All EDR fields listed in the **AssembledEDR** registry entry are included if the **L** segment has been received.

Upgrading Incomplete Calls to the New Container Description

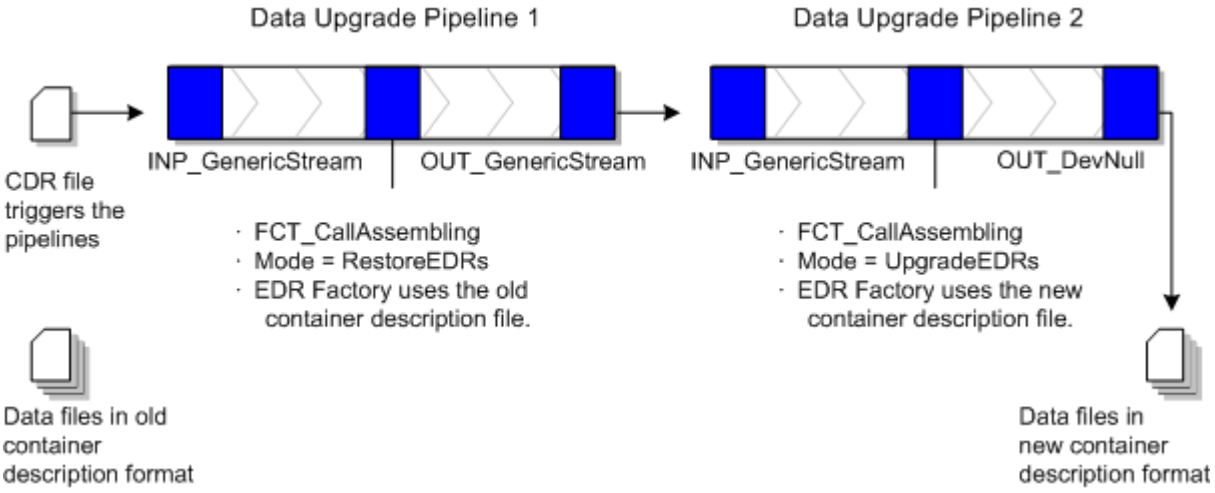
This section explains the steps necessary to update your partially-assembled call records to a new EDR container description. These tasks are necessary to correctly process any calls that are partially assembled at the time you shut down a pipeline to make the EDR container description changes.

> **Note:**
>
> The tasks in this section apply only to BRM implementations that use FCT_CallAssembling. If you use an external mechanism to assemble wireless records, the instructions in "Modifying and Loading the EDR Container Description" are all you need to modify your EDR container description.

To correctly rate the incomplete calls with your new EDR container description, you must first update the data file maintained by FCT_CallAssembling. The following procedure explains how to do this by running the EDRs through two miniature pipelines that you will create. After this conversion, you can process these calls through the pipeline set up to process call records using the new container description file. Before you run the pipelines, you will also create new input and output mapping and grammar files and a new stream file.

Figure 13-2 shows the process of upgrading stored EDRs to a new EDR container description.

Figure 13-2 Incomplete Call Description Update Process



> **Note:**
>
> Using the **RestoreEDRs** and **UpgradeEDRs** modes, FCT_CallAssembling does not assemble calls. Instead, it only migrates them from one format to another.

Follow these steps to update EDRs being stored by FCT_CallAssembling to your new EDR format. This procedure assumes that you have already created your new container description file. For details see "Modifying EDR Container Descriptions".

Run this entire procedure for each EDR container being changed.

> **Note:**
>
> If you are upgrading multiple pipelines to the new container description, you only need to create one set of data upgrade pipelines. You can use them to upgrade all FCT_CallAssembling pipelines getting the new container description.

1. Run the **pin_container_to_stream_format** utility on your old EDR container description file.

   For example:

   ```
   pin_container_to_stream_format -c oldcontainer.dsc -g OLD_ -m OLD_ -s OLD_
   ```

   This creates the stream, input and output grammar, and input and output mapping files that you will use to convert the partially assembled EDRs.

2. (Optional) Make any changes to the data fields in the new input grammar file to match your new EDR container description. The new container description file will have automatically supplied any necessary field *additions*. Edit the grammar file to specify any other changes to existing fields.

3. Create data upgrade pipeline 1. This pipeline will have only the following modules:

- INP_GenericStream using your old input mapping file (actually any input mapping file can be used here).
- FCT_CallAssembling. Set **Mode=RestoreEDRs** and set the **EdrFactory description** entry to your *old* container description file.
- OUT_GenericStream using the output mapping created by **pin_container_to_stream_format** in Step 1.

4. Create data upgrade pipeline 2. This pipeline will have only the following modules:
   - INP_GenericStream using the input mapping file created by **pin_container_to_stream_format** in Step 1 (modified as needed).
   - FCT_CallAssembling. Set **Mode=UpGradeEDRs** and set the **EdrFactory description** entry to your *new* container description file created in Step 1.
   - OUT_DevNull (removes the single CDR file you send in to initiate the pipelines).
5. Use a semaphore to stop the rating pipeline that is receiving the EDR container change. Wait for it to stop completely.
6. Back up the following files used by FCT_CallAssembling in your rating pipeline:
   - The most recent **.dat** file
   - All **.EDR** files
   - The most recent **.INDEX** file
7. Copy the most recent **.dat** file used by FCT_CallAssembling in your rating pipeline to the data directory used by FCT_CallAssembling in data upgrade pipeline 1 and data upgrade pipeline 2.
8. Copy the **.EDR** file used by FCT_CallAssembling from your rating pipeline to the EDR directory used by FCT_CallAssembling in data upgrade pipeline 1.
9. Copy the most recent **.INDEX** file used by FCT_CallAssembling in your rating pipeline to the index directory used by FCT_CallAssembling in data upgrade pipeline 1.
10. Use a semaphore to start the data upgrade pipelines.
11. Put one disposable CDR file in the input directory for data migration pipeline 1 to trigger the data upgrade. This file need only contain a single CDR, but it must use your old EDR container description format.

> **Note:**
>
> This CDR is needed only to start EDR processing; it will get dropped during processing.

12. Ensure that data upgrade pipeline processes have finished. The time required for these pipelines to update all EDRs varies with the number EDRs being updated.
13. Use a semaphore to stop the data upgrade pipelines.
14. Move the **.dat** and **.EDR** files from the data upgrade pipeline 2 output directory to the input directory of the FCT_CallAssembling pipeline receiving the EDR container change.
15. Install the new EDR container description in your rating pipeline, and start the pipeline.

This completes the process of upgrading all EDRs stored in FCT_CallAssembling while upgrading your EDR container description. These incomplete calls are processed normally.

About Serviceless Accounts as Charge Sharing Owners

Charge sharing group owners can be accounts that have no services. This enables you to bypass purchasing specific services for corporate accounts that are used exclusively to pool employee charges. For example, you can create a corporate account that assumes telephone, email, and IP charges for all employees in the marketing group. But, you do not have to purchase GSM, email, or IP services for that corporate account. If you set up a serviceless account as a charge sharing group owner, BRM applies the member charges to that account's default balance group.

Pipeline Manager uses the create, modify, and delete sharing group events to determine the owner of the sharing group. If the owner is an account without services, the PIN_FLD_SERVICE_OBJ value is NULL. If the owner is a service, the PIN_FLD_SERVICE_OBJ value is the POID of the service.

To create a charge sharing owner that is a serviceless account, perform the following:

1. Enable Pipeline Manager to accept serviceless accounts by setting the DAT_AccountBatch **LoadAccountForSharingOnly** registry entry to **True**. This prevents the pipeline from rejecting serviceless accounts if they are owners of balance element sharing groups. By default, this entry is set to **False.**
2. Create a serviceless account. First create a package that has no bundles. Then, purchase the package when you create the account. When you create a charge sharing group for this account, the result is an account-level charge sharing group not associated with any specific services.

Mapping Subscription Services in the Pipeline Manager Database

To rate subscription services in a pipeline, the data that defines the subscription service object must be mapped in the Pipeline Manager database. Sample mapping is provided by BRM for some service objects. However, if you set up a service object for the subscription service that is not already configured in the Pipeline Manager database, you must map that service to the data that defines it.

Use Pricing Center or PCC to map the subscription service in the following Pipeline Manager database tables:

- IFW_SERVICE specifies the pipeline rating service code to BRM rating service code mapping; for example, map **GSM** to **/service/telco/gsm**.
- IFW_SERVICE_MAP specifies the external data used to determine the internal service code. You define the mapping when you set up service code mapping.
- IFW_REF_MAP specifies the service-type-to-event-type mapping so that services are associated with usage events. This database table also specifies the prefix for the data that identifies an account; for example, MSISDN or IMSI.

> **Note:**
>
> Be sure to map a corresponding delayed event type to your subscription service object. For example, if you map to the usage event type **/event/session/telco/gsm**, also map to **/event/delayed/session/telco/gsm**. The delayed event type is used by Rated Event (RE) Loader when loading the rated events into the BRM database.
>
> If the delayed event type does not exist in the BRM database, you must create one. S

- IFW_ALIAS_MAP specifies which data to use for looking up an account in order to get the parameters for rating usage. You define the alias mapping when you set up EDR container fields.

  To define the alias mapping by using Pricing Center or PCC:

  1. From the **Pipeline Setup Toolbox**, select **EDR - EDR Container Description**.
  2. In the EDR Container Description dialog box, select the **ALL_RATE** sample container description and click **Edit**.
  3. Click the **Alias Map** tab and enter the alias data information for the subscription service.

Setting Up Batch Rating to Assign Items Based on Event Attributes

To set up Pipeline Manager to assign item tags to events based on event attributes:

1. Configure the following Pipeline Manager modules:
   - DAT_ItemAssign
   - FCT_ItemAssign
   - FCT_BillingRecord
2. Create a custom iScript that assigns item tags based on event attributes.
3. Load the rated events into the BRM database to update account balances and to create or update bill items.

Create a custom iScript that assigns item tags based on event attributes and fills in the DETAIL.ITEM_TAG field in the EDR container.

To enable your custom iScript to run in a pipeline, you must add an entry for it in the **wireless.reg** registry file. Configure this iScript to run *after* FCT_Account and *before* FCT_BillingRecord.

Sample registry entry:

```
# iScript to populate DETAIL.ITEM_TAG
#
IScript
{
  ModuleName = FCT_IScript
  Module
  {
   Active      = TRUE
```

```
    Source      = File
    Scripts
    {
     ItemTag
     {
      #iScript file that you created
      FileName = ./ISC_ItemTag.isc
     }
    }
   }
}
# end of iScript
```

Loading the Rated Events into the BRM Database

You use Rated Event (RE) Loader to load rated events into the BRM database. Before updating items in the database, RE Loader checks the updater flag in the RE Loader **Infranet.properties** file. If the flag value is **1**, RE Loader creates in the database the new item objects that were added. By default, the flag is set to **1**.

How Batch Rating Assigns Custom Bill Items

BRM batch rating uses the FCT_ItemAssign pipeline module to assign items based on event and service combinations and uses custom iScripts to assign items based on event attributes.

To assign items to events, Pipeline Manager performs these tasks:

1. During initialization, the DAT_ItemAssign module loads the specified **/config/item_types** object into memory and reserves a pool of POID IDs. If the information in the DAT_ItemAssign config object changes, you can use the DAT_itemAssign module's **Reload** semaphore to refresh the configuration changes.
2. Your custom iScript assigns an item tag based on event attributes to the DETAIL.ITEM_TAG EDR field.
3. FCT_ItemAssign calls DAT_ItemAssign with the item tag in the DETAIL.ITEM_TAG EDR field.
4. DAT_ItemAssign retrieves the item POID list from the DAT_AccountBatch module.
5. DAT_ItemAssign retrieves the item type for the given item tag from the **config/item_types** object in memory and searches the POID list for a matching item type.
6. If DAT_ItemAssign finds a matching POID, it returns that item POID (for example, **1 /item/new_york m m**) to FCT_ItemAssign.

   If DAT_ItemAssign does not find a matching POID, it creates a new POID ID from the POID pool it reserved and returns the new POID ID to FCT_ItemAssign.

   When the DAT_ItemAssign module creates new items, it updates DAT_AccountBatch with the new items it created; for example, **1 /item/new_york m m**.

   If the DETAIL.ITEM_TAG field is NULL, DAT_ItemAssign returns a default item POID from the item POID list.
7. The FCT_ItemAssign module assigns the item POID that it retrieves to the DETAIL.CUST_A.PRODUCT.SERVICE_USED_ITEM_POID EDR field for the product used for rating the event.
8. For sponsored usage events, the following occurs:

a. The FCT_BillingRecord module queries the DAT_ItemAssign module for items when required.

b. The DAT_ItemAssign module returns the pre-created items of type **/item/sponsor** to FCT_BillingRecord for sponsored events.

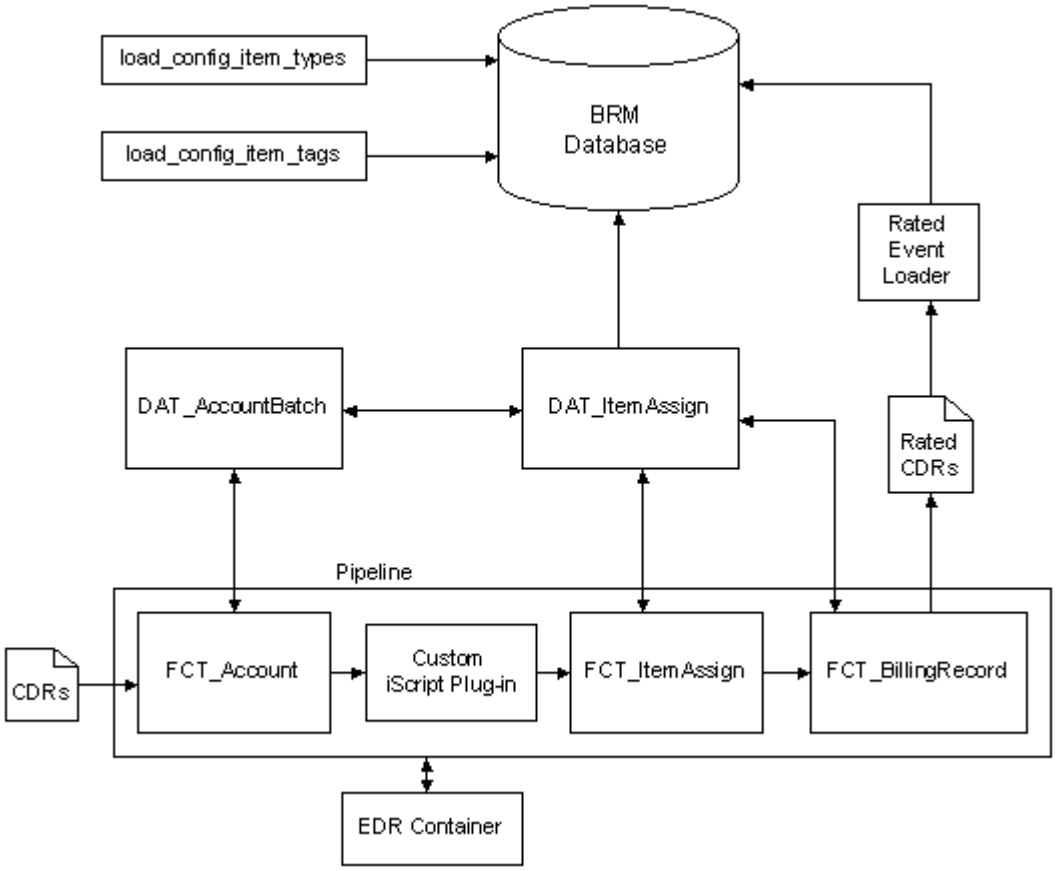
c. When the DAT_ItemAssign module creates new items, it updates DAT_AccountBatch with the new items it created; for example, **1 /item/sponsor m m**.

9. Pipeline Manager generates the rated events.

You use Rated Event (RE) Loader to load the rated events into the BRM database and to update the account balances and create or update bill items.

Figure 13-3 shows how items are assigned to events:

Figure 13-3 Assignment of Items to Events



How Batch Rating Assigns Custom Bill Items to Events for Balance Impacts

BRM batch rating uses the FCT_ItemAssign pipeline module to assign items based on event and service combinations and uses custom iScripts to assign items based on event attributes and balance impacts. The FCT_BillingRecord pipeline module converts impacts from charge offers (ChargePacket), discount offers (DiscountPacket), and taxes (TaxPacket) to balance impacts.

To assign custom bill items to events for balance impacts, Pipeline Manager performs the following tasks:

1. During initialization, the DAT_ItemAssign module loads the appropriate **/config/item_types** object into memory and reserves a pool of POID IDs. If the information in the DAT_ItemAssign config object changes, you can use the DAT_itemAssign module's **Reload** semaphore to refresh the configuration changes.
2. Your custom iScript does the following:
   a. Assigns an item tag based on event attributes to the DETAIL.ITEM_TAG EDR field.
   b. Changes the item POID for a specific balance impact in the event, using the following ChargePacket, DiscountPacket, and TaxPacket EDR tag container fields:
      - DETAIL.ASS.CBD.CP.ITEM_TAG
      - DETAIL.ASS.CBD.DP.ITEM_TAG
      - DETAIL.ASS.CBD.TP.ITEM_TAG
3. If the ChargePacket, DiscountPacket, and TaxPacket EDR container tag field values are NULL, FCT_ItemAssign calls DAT_ItemAssign with the item tag from the DETAIL.ITEM_TAG EDR field.

   If the ChargePacket, DiscountPacket, and TaxPacket EDR container tag field values are not NULL, FCT_ItemAssign calls DAT_ItemAssign with the item tag from the ChargePacket, DiscountPacket, and TaxPacket EDR container fields and the item tag from the DETAIL.ITEM_TAG EDR field.
4. DAT_ItemAssign retrieves the item POID list from the DAT_AccountBatch module.
5. DAT_ItemAssign retrieves the item type for the given item tag from the **config/item_types** object in memory and searches the POID list for a matching item type.
6. If DAT_ItemAssign finds a matching POID, it returns that item POID (for example, **1 /item/new_york m m**) to FCT_ItemAssign.

   If DAT_ItemAssign does not find a matching POID, it creates a new POID ID from the POID pool it reserved and returns the new POID ID to FCT_ItemAssign.

   When DAT_ItemAssign creates new items, it updates DAT_AccountBatch with the new items it created; for example, **1 /item/new_york m m**.

   If the DETAIL.ITEM_TAG field is NULL, DAT_ItemAssign returns a default item POID from the item POID list.
7. The FCT_ItemAssign module does the following:
   a. Assigns the item POID that it retrieves to the DETAIL.CUST_A.PRODUCT.SERVICE_USED_ITEM_POID EDR field for the charge offer used for rating the event.
   b. Assigns the item POID that it retrieves from ChargePacket, DiscountPacket, and TaxPacket EDR container tag fields respectively to:
      - DETAIL.ASS.CBD.CP.ITEM_POID
      - DETAIL.ASS.CBD.DP.ITEM_POID
      - DETAIL.ASS.CBD.TP.ITEM_POID

> **Note:**
>
> If no item tags are configured for ChargePacket, DiscountPacket, and TaxPacket, FCT_ItemAssign replaces the corresponding packet's item POID with the updated POID from DETAIL.CUST_A.PRODUCT.SERVICE_USED_ITEM_POID.

8. For sponsored usage events, the following occurs if the SplitSponsorItemByMember business parameter is enabled:
   a. Pipeline Manager receives input from the **config_item_tags.xml** and **config_item_type.xml** files to determine the sponsor item type.
   b. Assigns a type-only POID; for example, **/item/sponsor/usage-1**.
   c. The RE Loader assigns an appropriate sponsor item instance to the sponsor balance impacts.
9. Pipeline Manager generates the rated events.

You use RE Loader to load the rated events into the BRM database and to update the account balances and create or update bill items.

Creating a Batch Rating iScript for Balance Impacts

BRM batch rating uses custom iScripts to assign items based on event attributes.

Create a custom iScript that does the following:

1. Assigns an item tag based on event attributes to the DETAIL.ITEM_TAG EDR field.
2. Changes the item POID for a specific balance impact in an event, using the following fields in the EDR container:
   - DETAIL.ASS_CBD.CP.ITEM_TAG
   - DETAIL.ASS_CBD.DP.ITEM_TAG
   - DETAIL.ASS_CBD.TP.ITEM_TAG

To enable your custom iScript to run in a pipeline, you must add an entry for it in the **wireless.reg** registry file. Configure this iScript to run *after* FCT_Apply_Balance and *before* FCT_ItemAssign.

Verifying Item-Tag-to-Item-Type Mapping

You can generate a log file that contains the item-tag-to-item-type mapping information from the DAT_ItemAssign memory.

To generate a log file of the mapping:

1. Create a semaphore registry file with following entry:

```
ifw.DataPool.ItemAssignDataModule.Module.PrintData=TagTypeMap.txt
```

2. Copy the file into the semaphore directory. The default directory for semaphore files is *BRM_home***/ifw/semaphore**.

Pipeline Manager generates the **TagTypeMap.txt** file, which contains the tag and type mapping from the DAT_ItemAssign module memory.

For example, the file contains entries as follows:

```
Total number of Tag and Type Mapping entries: 3
------------------------------------------------------------
Tag : Type
------------------------------------------------------------
cycle_forward : /item/cycle_forward
misc : /item/misc
newyork: /item/newyork
```

14

Configuring Resource Rounding

This chapter describes how to configure balance impact rounding rules if you are using Oracle Communications Billing and Revenue Management (BRM) Pricing Center. This data is used by both real-time rating and pipeline batch rating.

Topics in this document:

> **Note:**
>
> If you are using Oracle Communications Pricing Design Center (PDC) to create product offerings, see "Configuring Balance Impact Rounding" in *PDC Creating Product Offerings*.

About Resource Rounding

BRM enables you to create various rounding rules for different resources, events, and processes such as rating and discounting. You create rounding rules for several reasons:

- To increase the accuracy of rating and discounting results.
- To process usage fees more efficiently. For example, an infinite number such as 5.333... is more easily processed when it is rounded.
- To round for various currencies that use a different number of digits to the right of the decimal (for example, 10.25 dollars and 10 yen).
- To comply with currency conversion rules.
- To bill customers an amount that they can actually pay.

You configure rounding by editing the **pin_beid** file and loading the contents of the file into the **/config/beid** object in the BRM database. For more information, see "Configuring Resource Rounding".

About Rounding Rules

You can configure resource rounding based on the following rounding criteria:

- The *rounding scale* is the number of significant digits to the right of the decimal point. For example, a scale of 2 applied to 10.321111 rounds to 10.32.
- The *rounding mode* defines whether the number is rounded up, down, or not at all, based on the value of the digit following the last significant digit. See "About Rounding Modes".
- The process to which the rounding configuration applies. You can specify one of these processes: rating, discounting, taxation, and accounts receivable (A/R). A/R includes actions such as billing, payments, adjustments, cancellations, and G/L reporting.

Specifying the process enables you to round differently based on the operation. For example, you can round up using six decimal places for rating and round down using two decimal places for billing.

- The event type to which the rounding configuration applies. This enables you to round differently for events that represent specific types of usage, cycle fees, discounts, and rollovers. For example, US dollars and purchase events (**/event/purchase**).

Table 14-1 show how a US dollars resource and purchase event combination can be rounded various ways for various processes:

Table 14-1 Currency Resource and Event Type

For a US Dollars Resource	For This Process	Use This Rounding Scale	Use This Rounding Mode
Event type = **/event/billing/product/fee/purchase**	Rating	6	Down
Event type = **/event/billing/product/fee/purchase**	Discounting	6	Up
Event type = **/event/billing/product/fee/purchase**	A/R	2	Nearest
Event type = **/event/billing/product/fee/purchase**	Taxation	2	Nearest

You can specify any combination of scale and mode for resource, event, and process combinations.

After each process that performs rounding (rating, discounting, and so forth), the balance impact of the event contains the rounded amount.

To configure rounding, you specify the rounding rules in the resource configuration file (*BRM_Home*/**sys/data/pricing/example/pin_beid**), and then you load the file by using the **load_pin_beid** utility. The **pin_beid** file contains default rounding rules that you can use. For more information, see "Configuring Resource Rounding".

> **Note:**
>
> To round noncurrency aggregation counter balances for discounting purposes, use Pipeline Manager discount expressions. For more information, see "About Rounding Aggregation Counter Resources for Discounting".

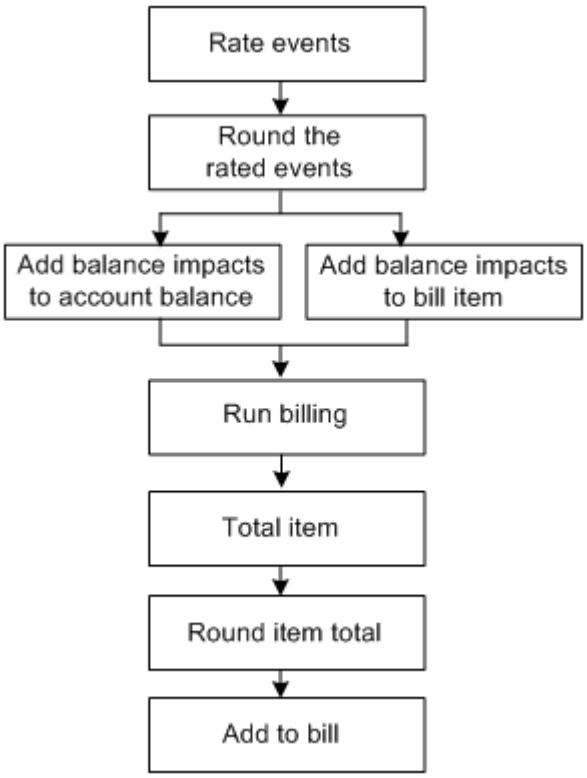
How BRM Applies Rounding

Rating, discounting, and taxation produce balance impacts, which are rounded and applied to the customer's account balance. The balance impacts are rounded to the scale and mode configured for the event, resource, and process combination.

Balance impacts for an account are stored in bill items, which are associated with the customer's bill. When you run billing, the item totals are rounded according to the A/R rounding rule. Item totals are rounded before being added to the bill so that the bill itself never needs rounding. Before billing, item totals are unrounded and their amounts reflect the scale configured for the associated events.

Figure 14-1 show the general process of balance impact rounding:

Figure 14-1 BRM Balance Impact Rounding



Both real-time rating and pipeline batch rating perform rounding. During pipeline rating, the pipeline rounds the amount in each charge and discount packet.

To round in a pipeline, you configure the Pipeline Manager rounding module. See "Setting Up Rounding in Pipeline Manager". To configure the rounding rules, see "Configuring Resource Rounding".

Assuming that rounding is configured for all events and all processes (rating, discounting, taxation, and A/R), rounding is performed in the following order:

1. **Purchase and cycle fees**. When a plan is purchased, the purchase and cycle fees are rounded, if necessary, and a balance impact is generated with the rounded amount. Purchase and cycle fees typically only need rounding when a user purchases or cancels a plan in the middle of the billing cycle. In this case, cycle fees and noncurrency grant amounts are prorated, if necessary, and then rounded.
2. **Usage fees**. As customers use their services, the usage events are rated, the usage fees are rounded, and balance impacts are generated with the rounded amounts.
3. **Discount fees**. If a usage discount applies, the discount is calculated on the rounded usage fee, and then the discount is rounded and a discount balance impact is generated with the rounded amount. If there are multiple usage discounts, each discount is rounded in sequence.
4. **Taxes**. If tax should be applied to the event, the tax is calculated on the fee rounded from the previous stage, and then the tax is rounded and a balance impact is generated with the rounded amount.

5. **Billing discounts**. When you run billing, if a billing-time discount applies, it is applied to the total of the rounded usage items, and then the discount is rounded and a discount balance impact is generated with the rounded amount.
6. **Bills**. When the bill is generated, the total amount in each item is rounded, and then all item totals are summed and included in the bill.
7. **Rollover**. If there are any rollover amounts, the rollover is calculated on the total rounded usage, and then the rollover amount is rounded and applied to the next billing cycle. For example, if 100 minutes can be rolled over and the customer used 60.4 minutes, the unused amount of 39.6 minutes is rounded and then added to the account balance for the next cycle.

   To round for rollover, you configure a rounding rule for the noncurrency resource and cycle forward event combination. Rollover is typically rounded up.

About Rounding and A/R Actions

When entering amounts for A/R actions such as adjustments and refunds, customer service representatives (CSRs) typically use *natural scale*: that is, the scale commonly used in the marketplace for that resource. For example, a person who purchases a book using US dollars cannot make change smaller than one cent (.01 dollars), making 2 the natural scale for US dollars. However, if a CSR reverses or adjusts an event before billing, the scale used is the one in the event's balance impact. By default, this is the natural scale, unless you change this to use a scale other than natural.

About Rounding Billed and Unbilled Items

The balance impacts of events associated with items can have a high scale. Before billing, item totals reflect the scale of their associated events. During billing, item totals are rounded using the A/R rounding configuration so that customers' bills display the natural scale for the resource. However, the events associated with billed items still retain their pre-billing scale.

If any operation is performed on billed items: for example, event-level and item-level adjustments, BRM rounds the balance impact of the operation according to the A/R rounding rule to maintain G/L integrity. Because actions on billed items are rounded when they occur, only pending items are rounded when billing is run.

About Rounding for Specific Event Types

When you configure a rounding rule for a specific event type, such as cycle or session events, that rounding rule applies to those events only for the process specified. If you do not configure a rule for every process, all other processes for that event type use the default rounding rule defined in the **pin_beid** file. The default rounding rule specifies natural scale for all events and the rounding mode most commonly used for the process. A rule for all events is specified by using an asterisk (*****) as the default event type.

For example, given the configuration in Table 14-2, during rating, session events are rounded down and have a scale of 6. During taxation, however, session events use the default rule of rounding to the nearest with a scale of 2:

Table 14-2 Rounding for Specific Event Types

Resource	Event Type	Process	Rounding Scale	Rounding Mode
US dollars	**/event/session**	Rating	6	Down
US dollars	***** (all events)	Taxation	2	Nearest

About Rounding Aggregation Counter Resources for Discounting

For discount rounding, the rounding configurations in the **pin_beid** file are used to round only the balance impacts of discounting, not input balances such as aggregation counters. To round aggregation counter balances in a pipeline, you use pipeline discount expressions when you configure your discounts. To configure aggregation counter rounding, see "Configuring Rounding Rules for Aggregation Counter Resources".

About G/L Report Rounding

G/L report rounding uses the rounding rule configured for A/R actions. The rounded totals in G/L reports might differ slightly from the total of the rounded bills. This is because item totals are rounded for billing, and journal entries are rounded for G/L reports. Also, G/L reports are rounded differently before and after billing. For more information, see "About Rounding and G/L Reports" in *BRM Collecting General Ledger Data*.

If there is any difference between the rounded journal entries in G/L reports and the rounded bill items, BRM records the difference in the G/L. You configure a G/L ID for the rounding difference and configure BRM to record the differences. See "Configuring to Record Rounding Differences in the G/L".

About Rounding Modes

A rounding mode defines whether a number is rounded up, down, or not at all. The rounding mode rounds to the specified scale. For example, using a scale of 2, rounding up 10.2369 results in 10.24. If the scale is 3, rounding up results in 10.237.

The following rounding mode values that you can use in the **pin_beid** file are defined in the *BRM_Home***/include/pin_bill.h** file:

- 0: PIN_BEID_ROUND_NEAREST

  This mode rounds up or down depending on the value of the digit following the last significant digit. If the additional digit is 0-4, the last significant digit remains the same. If the additional digit is 5-9, the last significant digit is rounded up. For example, if the scale is 2, 10.144 rounds to 10.14 and 10.145 rounds to 10.15. This is the most common rounding method.

- 1: PIN_BEID_ROUND_UP

  This mode rounds up when the digit following the last significant digit is greater than 0. For example, If the scale is 2, 10.151 rounds to 10.16. If the scale is 1, it rounds to 10.2.

- 2: PIN_BEID_ROUND_DOWN

  This mode truncates all digits following the last significant digit. For example, if the scale is 2, 10.159 rounds to 10.15. If the scale is 1, it rounds to 10.1.

- 3: PIN_BEID_ROUND_EVEN

  This mode rounds one of three ways depending on the value of the digit following the last significant digit:

  - If it is less than 5, truncate all digits following the last significant digit.
  - If it is greater than 5, round up.
  - If it is 5, round to the nearest even digit. For example, if the scale is 2, 10.155 and 10.165 both round to 10.16 because 6 is an even number.

- 4: PIN_BEID_ROUND_FLOOR

  This mode rounds numbers toward a negative value (that is, the rounded number is always less than the unrounded number). This enables you to round balance impacts so that customers always benefit. For example, if the scale is 2, a credit to a customer of -7.999 is rounded to -8.00, and a debit of 7.999 is rounded to 7.99.

- The following two modes perform the same rounding as their non-alternative counterparts (ROUND_FLOOR and ROUND_DOWN), except that they compensate for possible loss of precision when rounding down by first rounding with a mode of NEAREST using a scale that is two digits greater than the scale you configure.

  – 5: PIN_BEID_ROUND_FLOOR_ALT

  – 6: PIN_BEID_ROUND_DOWN_ALT

  For more information, see "About Rounding Modes That Correct for Loss of Precision".

About Rounding Modes That Correct for Loss of Precision

Some calculations produce results that are slightly less than expected when a value is rounded down. For example, when BRM prorates a $60.00 cycle fee for 20 out of 30 active days, the calculation is (20/30) * $60.00. The expected result is a fee of $40.00. However, because 20/30 evaluates to 0.666..., when this is multiplied by 60 and rounded down, the actual result is a fee of $39.99.

BRM provides two alternative rounding modes that compensate for possible precision loss when rounding down: ROUND_DOWN_ALT and ROUND_FLOOR_ALT. These modes perform the same rounding as their non-alternative counterparts (ROUND_DOWN and ROUND_FLOOR) after first compensating for loss of precision.

> **Note:**
>
> ROUND_DOWN_ALT and ROUND_FLOOR_ALT are not supported in the BRM PCM Java API and in discount expressions.

When these modes are used, if a decimal should be rounded down, BRM performs two rounding functions: The decimal is rounded by using the ROUND_NEAREST rounding mode and a scale that is two more than the scale that you request. It is then rounded down.

For example, if you configure the rounding mode as ROUND_DOWN_ALT and a rounding scale of 2, and the decimal to round is 7.999..., BRM truncates the infinite decimal to the system maximum and then rounds this decimal to the nearest using a scale of 4 (2 more than the configured scale of 2), which results in 8.0000. This decimal is then rounded down using the configured scale of 2, resulting in 8.00 as shown in Table 14-3:

Table 14-3 Rounding Modes That Correct for Precision

Do This	Why
7.999...	Original decimal
7.99999999999999	Truncated to the system max
8.0000	ROUND_NEAREST, using requested scale + 2
8.00	ROUND_DOWN to the requested scale of 2

Had the original decimal of 7.999... not been rounded to the nearest first and only rounded down, the result would be 7.99.

To compensate for possible loss of precision, the alternative rounding modes consider two decimal places more than the non-alternative rounding modes. Therefore, the greatest amount that will be modified by using the alternative rounding modes, and still compensate for loss of precision, is less than the greatest amount that will be modified by using the non-alternative rounding modes.

When Rounding Is Not Applied

When the requested rounding scale is greater than the scale of the number being processed, rounding is not required. This occurs when:

- You configure a rounding scale equal to or greater than the maximum number of digits allowed by the system. For example, if the system allows 15 digits and you set the rounding scale to 15 or greater, rounding has no effect.
- You call a rounding function and request a scale that is equal to or greater than the current scale of the decimal: for example, when the decimal is 10.89766 and the scale requested is 5 or greater.
- A computation or expression results in a decimal with a scale that is equal to or less than the configured or requested scale: for example, when a computation results in the decimal 1.98 and the configured scale is 2 or greater.

Configuring Resource Rounding

To set up rounding, perform the following tasks:

- Configuring Rounding Rules
- Configuring Rounding Rules for Aggregation Counter Resources
- Configuring to Record Rounding Differences in the G/L
- Setting Up Rounding in Pipeline Manager

About Configuring Rounding Rules

You define rounding rules in the balance element ID (BEID) configuration file (B*RM_Home*/**sys/data/pricing/example/pin_beid**) and then run the **load_pin_beid** utility to load the contents of the file into the **/config/beid** object in the BRM database. See "Configuring Rounding Rules".

The **pin_beid** file contains default rounding rules based on the most commonly used rounding for the resource and operation. The default rules specify a configuration for all events by using an asterisk (*****) as the default event type. For an explanation of the configuration syntax and fields, see the **pin_beid** file.

You configure rounding for resource and event type combinations (for example, US dollars and **/event/session/telco/gsm** events). For each resource and event combination, you specify a rounding scale, a rounding mode, and the process for which this rule applies. For the process, you specify one of these enumerated values in the **pin_beid** file:

- Rating = **0**
- Discount = **1**
- Taxation = **2**

- A/R = **3**

To configure an event and resource rule for every process, you add an entry for each process.

> **Note:**
>
> - You do not need to configure rounding for all of the processes. However, you must configure rounding for every resource and event type that has a balance impact.
> - If you add resources that are not already in the **pin_beid** file, you should always include a default rounding configuration for that resource that applies to all event types. Any event type that is not explicitly specified for the resource uses this default rounding rule.
> - You can use a single asterisk (*****) to denote a default value such as any event type. However, in combination with other characters, you must use valid regular expressions. For example, specifying **/event/*** is incorrect; specifying **/event/(.)*** is correct.

> **Note:**
>
> You can use regular expressions in the rounding configurations. For example, **/event/session/(.)*** matches all session events.

For more information, see "Configuring Rounding Rules"**.**

Prioritizing Rounding Rules

BRM applies the first matching rounding rule in the **/config/beid** file. Therefore, if you have more than one rule that matches an event, resource, and process combination, add them in order of priority in the **pin_beid** file. For example, a configuration that matches a specific event type should be added before a configuration that matches all event types.

If there is no rounding rule in the **/config/beid** object that matches the resource, event, and process combination, and no default rule that applies to all events, the event is not rounded.

About Rounding Mode Values

The rounding modes you specify in the **pin_beid** file have corresponding modes in the following API components. However, their values are not all the same:

- **The BRM decimal data type functions**. When BRM invokes a decimal data type function, it converts the rounding mode in the **/config/beid** object into the corresponding rounding parameter used by the decimal data type function. If you call decimal data type functions in custom applications, specify the function's rounding mode, not the BEID rounding mode.
- **The BAS Decimal class**. The FCT_Rounding pipeline module converts the rounding mode in the **/config/beid** object into the corresponding rounding mode used by the BAS rounding method in Pipeline Manager. If you use the BAS Decimal rounding method in custom pipeline modules and iScripts, you should include the FCT_Rounding module in your pipeline. Otherwise, you will need to convert the rounding mode in the **/config/beid** object into the BAS rounding mode before calling the BAS rounding method.

Table 14-4 shows the BEID rounding modes and the corresponding rounding parameters for the decimal data type functions and the BAS rounding modes:

Table 14-4 BEID Rounding Modes

BEID Rounding Mode (specified in pin_bill.h)	Rounding Mode Parameter for pbo_decimal functions	BAS Decimal Rounding Mode
0: PIN_BEID_ROUND_NEAREST	5: ROUND_HALF_UP	0: PLAIN
1: PIN_BEID_ROUND_UP	1: ROUND_UP	1: UP
2: PIN_BEID_ROUND_DOWN	2: ROUND_DOWN	2: DOWN and TRUNCATE
3: PIN_BEID_ROUND_EVEN	7: ROUND_HALF_EVEN	3: BANKERS
4: PIN_BEID_ROUND_FLOOR	4: ROUND_FLOOR	101: FLOOR
5: PIN_BEID_ROUND_FLOOR_ALT	8: ROUND_FLOOR_ALT	103: FLOOR_ALT
6:PIN_BEID_ROUND_DOWN_ALT	9: ROUND_DOWN_ALT	102: DOWN_ALT
N/A	3: ROUND_CEILING	N/A
N/A	6: ROUND_HALF_DOWN	N/A
N/A	10: ROUND_UNNECESSARY	N/A

Configuring Rounding Rules

To configure rounding rules, you edit the BEID configuration file and then run the **load_pin_beid** utility to load the contents of the file into the **/config/beid** object in the BRM database.

> **Note:**
>
> The **load_pin_beid** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

To configure resource rounding:

1. Edit the **pin_beid** file in *BRM_Home***/sys/data/pricing/example**. The **pin_beid** file includes instructions.

   > **Note:**
   >
   > The **load_pin_beid** utility overwrites the existing resources. If you are updating resources, you cannot load new resources only. You must load complete sets of resources each time you run the **load_pin_beid** utility.

2. Save the **pin_beid** file.
3. Use the following command to run the **load_pin_beid** utility:

   ```
   load_pin_beid pin_beid
   ```

   If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

```
load_pin_beid BRM_Home/sys/data/pricing/example/pin_beid
```

> **Note:**
>
> If you copy the **pin_beid** file to the directory from which you run the **load_pin_beid** utility, you do not have to specify the path or file name. The file must be named **pin_beid**.

For more information, see "load_pin_beid".

4. Stop and restart the Connection Manager (CM).

To verify that the network elements were loaded, you can display the **/config/beid** object by using Object Browser, or use the **robj** command with the **testnap** utility.

After loading the **pin_beid** file, functions that calculate fees and discounts use the rounding rules in the **/config/beid** object.

Configuring Rounding Rules for Aggregation Counter Resources

You use Pricing Center to configure rounding rules for aggregation counter resources. When you set up your discount configurations, use discount expressions to specify how to round the resource. Use the following discount expression syntax:

```
round(expression, rounding_scale, rounding_mode)
```

where expression defines the resource balance to round. This can be any discount expression. To round an aggregation balance, use the **Bal** expression. For more information, see the discussion about setting up discounts in the Pricing Center Help.

For example, to round a balance down to two decimal places for an aggregation counter resource with ID 100099, use the following expression:

```
round( Bal(1000099), 2, ROUND_DOWN )
```

About Rounding Modes for Discount Expressions

Rounding modes for discount expressions are equivalent to those you specify in the **pin_beid** file, but they have slightly different names. Table 14-5 lists the rounding mode values you can specify in discount expressions and their **pin_beid** file counterpart.

Table 14-5 Rounding Modes for Discount Expressions

Discount Expression Rounding Mode	Rounding Mode Used in the pin_beid File
ROUND_PLAIN	Nearest
ROUND_UP	Up
ROUND_DOWN	Down
ROUND_BANKERS	Even

For a definition of what these modes represent, see "About Rounding Modes".

Configuring to Record Rounding Differences in the G/L

To record any difference between rounded bill items and the rounded total in the G/L, perform the following tasks:

- Defining a G/L ID for Rounding Differences
- Mapping the Rounding G/L ID to an Event
- Configuring BRM to Record Rounding Differences

For information about how rounding is performed in G/L reports, see the discussion about rounding and G/L reports in *BRM Collecting General Ledger Data*.

Defining a G/L ID for Rounding Differences

You define a G/L ID for rounding to include the rounding difference in G/L reports so that they can be accurately reconciled.

To define G/L IDs, you edit the G/L ID configuration file and then run the **load_pin_glid** utility to load the contents of the file into the **/config/glid** object in the BRM database.

> **Note:**
>
> The **load_pin_glid** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

To define a rounding G/L ID:

1. If necessary, edit the G/L ID configuration file in *BRM_Home***/sys/data/pricing/example/pin_glid**. If the following entry is not present, add it:

```
#====================================================================
# G/L ID for rounding adjustments
#====================================================================
glid
id 1512
descr Rounding Epsilon
gl_acct billed         gross    rounding.debit    rounding.credit
gl_acct billed         net      rounding.debit    rounding.credit
gl_acct billed         disc     rounding.credit   rounding.debit
gl_acct billed_earned  gross    rounding.debit    rounding.credit
gl_acct billed_earned  net      rounding.debit    rounding.credit
gl_acct billed_earned  disc     rounding.credit   rounding.debit
gl_acct unbilled       gross    rounding.debit    rounding.credit
gl_acct unbilled       net      rounding.debit    rounding.credit
gl_acct unbilled       disc     rounding.credit   rounding.debit
gl_acct unbilled_earned gross   rounding.debit    rounding.credit
gl_acct unbilled_earned net     rounding.debit    rounding.credit
gl_acct unbilled_earned disc    rounding.credit   rounding.debit
)
```

> **Note:**
>
> The **load_pin_glid** utility overwrites existing G/L IDs. If you are updating G/L IDs, you cannot load new G/L IDs only. You must load complete sets of G/L IDs each time you run the **load_pin_glid** utility.

2. Save and close the file.
3. Use the following command to run the **load_pin_glid** utility:

   ```
   load_pin_glid pin_glid_file
   ```

For more information, see the discussion about loading general ledger configuration data in *BRM Collecting General Ledger Data*.

Mapping the Rounding G/L ID to an Event

Because the rounding difference is not a rated event, you must map the G/L ID to an event type. G/L ID mapping is defined in the **reasons.***locale* file. You can find a sample of this file in the *BRM_Home***/sys/msgs/reasoncodes** directory. The sample file is named **reasons.en_US** and contains the following default entry for the rounding G/L ID mapping:

```
DOMAIN = "Others" ;
STR
     EVENT-GLID
          . . .
          /event/journal/epsilon"        1512 ;
     EVENT-GLID-END
```

> **Note:**
>
> **/event/journal/epsilon** is a dummy event type used for reference only.

To change the G/L ID for rounding, you must edit and reload the file. The G/L ID you define in the **reasons.***locale* and **pin_glid** files must match. See "Configuring to Record Rounding Differences in the G/L".

To map the G/L ID for rounding to an event, you use the **load_localized_strings** utility to load the contents of the file into the **/config/map_glid** object. When you run the **load_localized_strings** utility, use this command:

```
load_localized_strings reasons.locale
```

> **Note:**
>
> If you are loading a localized version of this file, use the correct file extension for your locale.

> **Note:**
>
> The **load_localized_strings** utility overwrites the existing G/L ID maps. If you are updating this object, you cannot load new G/L ID maps only. You must load complete sets of G/L ID maps each time you run the **load_localized_strings** utility.

For information on loading the **reasons.***locale* file, see the discussion about creating a localized version of BRM in *BRM Developer's Guide*.

Configuring BRM to Record Rounding Differences

By default, rounding differences are not recorded in G/L reports. You can enable this feature by modifying a field in the **billing** instance of the **/config/business_params** object.

You modify the **/config/business_params** object using the **pin_bus_params** utility.

To enable BRM to record rounding differences:

1. Use the following command to create an editable XML file from the **billing** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for the following line:

   ```
   <GenerateJournalEpsilon>disabled</GenerateJournalEpsilon>
   ```

3. Change **disabled** to **enabled**.

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

4. Save and close the file.

5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

   You should run this command from the *BRM_Home***/sys/data/config** directory, which includes support files used by the utility.

6. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

7. Stop and restart the Connection Manager (CM).

8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see the discussion about **pin_multidb** in *BRM System Administrator's Guide*.

Setting Up Rounding in Pipeline Manager

Pipeline Manager uses the rounding rules specified in the **/config/beid** object when applying rounding. To round balance impacts in Pipeline Manager, you configure the FCT_Rounding module. See "About Configuring the FCT_Rounding Module".

You can also set up separate rounding rules when defining pricing. See "About Rounding Rules in Pricing".

Rounding rules can conflict. For more information, see "Avoiding Rounding Conflicts in Pipeline Manager".

About Configuring the FCT_Rounding Module

The FCT_Rounding module finds the rounding rule in the **/config/beid** object based on the event, balance element, and process (rating, discounting, or taxation) combination and then rounds the amount in the relevant EDR packet.

Add FCT_Rounding to the pipeline after the processing module for which it is rounding:

- **To round for rating**, add this module after the FCT_RateAdjust module.
- **To round for taxation**, add this module after the ISC_TaxCalc iScript module.
- **To round for real-time discounting**, add this module after the FCT_Discount module.
- **To round for batch discounting**, add this module after the FCT_Discount module and before the FCT_ApplyBalance module.

You specify the process for which this module is rounding in the **Mode** entry of the module registry:

- **Rating** = Round the balance impact of rating.
- **Taxation** = Round the balance impact of taxation.
- **Discounting** = Round the balance impact of discounting.

For more information, see "Avoiding Rounding Conflicts in Pipeline Manager".

About Rounding Rules in Pricing

You specify one of these rounding methods when defining pricing:

- **None**: Only the rounding rules configured in FCT_Rounding are used.
- **Plain**: Round up if the last significant digit is 5 or greater.
- **Up**: Always round up to the next highest digit.
- **Down**: Always round down to the next lowest digit.
- **Bank**: If the last significant digit is 5, make it even.

If you specify a rounding method other than **None** for pricing, Pipeline Manager will use that rounding method during rating.

For more information, see "Avoiding Rounding Conflicts in Pipeline Manager".

Avoiding Rounding Conflicts in Pipeline Manager

For Pipeline Manager, it is possible to specify rounding rules in two places that apply to the same event. Pipeline Manager uses different rounding rules at different points in the rating process:

- The rounding method specified for the pricing is used by the rating module.
- The rounding method specified for the balance element and event is used in the FCT_Rounding module, which is usually run after the FCT_RateAdjust and FCT_Discount modules in the pipeline.

Because rounding can take place at these different points in the pipeline, you can get unexpected results. To avoid conflicts between different rounding rules, you can do one of the following:

- Choose **None** for the rounding method in the pricing. If you do this, only the rounding rules configured in FCT_Rounding are used.
- If you select a rounding method other than **None** in the pricing, ensure that the selected method is consistent with the rounding method that FCT_Rounding will use.

Rounding Examples

Rounding is performed after rating, discounting, taxation, and A/R actions such as billing and adjustments.

Table 14-6 shows the resulting balance impacts of rounded charges for rating, discounting, taxation, and billing. This example rounds to the nearest mode and uses these scales:

- 2 for purchase events, taxation, and A/R
- 5 for rating and discounting

Table 14-6 Rounding Examples

Action / Process	Calculated Charge	Rounded Charge & Balance Impact	Account Balance
Purchase plan Rate cycle fee	9.95	9.95	9.95
Use service Rate usage	5.23456789	5.23457	15.18456
Discount usage fee 10%	0.523456789	0.52346	14.66111
Tax usage event 3% (after discount)	0.1413333 = 3% * (rounded usage fee - rounded discount)	0.14	14.80111
Run billing Apply billing-time discount of 5% off total usage	0.24250... = 5% * usage item total after A/R rounding (The usage total does not include the cycle fee.)	0.2425	14.55861

Table 14-6 (Cont.) Rounding Examples

Action / Process	Calculated Charge	Rounded Charge & Balance Impact	Account Balance
Create bill	14.55861 = Total of all items	14.56 (No balance impact)	14.56

Correcting for Precision Loss When Rounding Down

This example shows the results of rounding when you use the ROUND_DOWN_ALT and ROUND_FLOOR_ALT modes. For more information, see "About Rounding Modes That Correct for Loss of Precision".

The ROUND_DOWN_ALT and ROUND_FLOOR_ALT modes produce different results than ROUND_DOWN and ROUND_FLOOR only when the three digits following the last significant digit are 995 or greater. (The last significant digit is the digit in the decimal place corresponding to the scale: If the scale is 2, the last significant digit in the number 1.23456 is 3.)

For example:

Table 14-7 shows some rounding results of the ROUND_DOWN_ALT and ROUND_FLOOR_ALT modes as compared to their non-alternative rounding counterparts (ROUND_DOWN and ROUND_FLOOR) for various decimal values and rounding scales.

Table 14-7 Rounding Results

Decimal	Scale	Rounding Mode	Rounding Mode	Rounding Mode	Rounding Mode
NA	NA	DOWN	DOWN_ALT	FLOOR	FLOOR_ALT
1.5256	2	1.52	1.52	1.52	1.52
-1.5256	0	-1	-1	-2	-2
12.8999...	0	12	12	12	12
12.8999...	1	12.8	12.9	12.8	12.9
12.8999...	2	12.89	12.90	12.89	12.90
-12.8999...	1	-12.8	-12.9	-12.8	-12.9
-12.8999...	2	-12.89	-12.900	-12.89	-12.90
-6.9990	2	-6.99	-6.99	-7.00	-7.00
-6.9990	3	-6.999	-6.999	-6.999	-6.999
7.999...	0	7	8	7	8
7.999...	1	7.9	8.0	7.9	8.0
7.999...	2	7.99	8.00	7.99	8.00
-7.999...	0	-7	-8	-8	-8
-7.999...	2	-7.99	-8.00	-8.00	-8.00

Rounding Using Different Modes

The aggregated effects of rounding on the final balance impact is determined by the mode and scale that you configure. The higher the scale, the less effect the rounding mode has on the final balance impact.

For example, Table 14-8 shows the impact of various rounding mode combinations for rating a usage fee of $1.1234567 that includes a 10% discount. Both rating and discounting use a scale of 6:

Table 14-8 Rounding Modes

Processing Stage (Rating and Discounting)	Rating: Round Down Discounting: Round Down	Rating: Round Down Discounting: Round Up	Rating: Round Up Discounting: Round Down	Rating: Round Up Discounting: Round Up
Round for rating	1.123456	1.123456	1.123457	1.123457
Calculate 10% discount	.1123456	.1123456	.1123457	.1123457
Round discounted amount	.112345	.112346	.112345	.112346
Apply discount to usage fee to get final balance impact	1.011111	1.011110	1.011112	1.011111

In this example, the difference in the final balance impacts is small because the scale is high and probably will not change the final amount on the bill. However, when many events are summed in an item, or the scale is small, such as 2 or 3, the differences become greater.

> **Note:**
>
> If you calculate the discount without rounding the event, the configuration where rating is rounded down and discounting is rounded up returns the most accurate result. Therefore, this is the best mode configuration to use when you discount events.

Modifying a Rounding Rule

To modify a rounding rule, you must change the rule in the **pin_beid** file and reload the file by running the **load_pin_beid** utility. See "Configuring Resource Rounding".

15

Managing Sub-Balances

This document describes how to manage sub-balances in Oracle Communications Billing and Revenue Management (BRM) Pricing Center. It describes balance impacts, credit limits, and rollovers.

Topics in this document:

About Sub-Balances

Each balance in a balance group can include one or more sub-balances. A balance includes sub-balances when portions of the balance are valid at different times or when a portion of the balance is a loan. For example, a balance of minutes might include 300 minutes that are valid only for the current month and 1000 minutes that never expire.

A currency sub-balance can store the balances for multiple services. For example, an account that owns two products that cost $25.00 per product has a starting currency sub-balance of $50.00, providing the services are associated with the same balance group and have the same validity period.

A currency balance can also store sub-balances for loans. For example, if an account subscribed to a monthly package that costs $25.00 only has a balance of $15.00 at billing time, you can grant a loan for the remaining $10.00. The balance would then consist of two sub-balances; a $15 regular sub-balance and a $10 loan sub-balance.

Sub-balances include the following information:

- The sub-type. This indicates whether the sub-balance is a loan or not.
- The start time and end time for which the sub-balance is valid.
  Common resources with the same validity periods are stored in the same sub-balance. Resources with unique validity periods are stored in separate sub-balances. For more information, see "How Resources in Validity-Based Sub-Balances Are Updated".

  For information about rounding the start times of cycle and purchase grants to midnight, see "Configuring Time-Stamp Rounding for Validity Period Start Times".
- The current amount of the sub-balance.

- Consumed reservation (or active reserved amount) which tracks the consumed reservation and is used to set up notifications to the network.
  See the discussion on calculating reservation balances in *BRM Telco Integration*.
- The fields in the event record or object (referred to as “contributors") that contribute to how sub-balances are created, updated, and retrieved. For example, to retrieve the total available balance for a specific service, the service object is specified. To deduct free minutes for a phone call, the session object is specified. A separate sub-balance is kept for each unique contributor. See "About Sub-Balance Contributors".
- Rollover data such as the rollover period and the resource amount that is rolled over, if any. For more information, see "About Rollovers".
- The ID of the product or discount that granted the resource (referred to as “grantor object.")

You can configure sub-balances to track various types of resources and usage. See "About Configuring Sub-Balances".

About Noncurrency Sub-Balances

A noncurrency sub-balance typically has a limited validity period (for example, the period during which free minutes can be used). Noncurrency sub-balances can contain various types of resources, such as:

- Free minutes.
- Frequent flyer miles.
- Loyalty points.
- Number of emails or text messages.

A noncurrency sub-balance can also keep track of the total resources used for discounts that are shared among several accounts. In this case, the sub-balance acts as a counter to keep track of the total consumed resource.

When granting a noncurrency resource, if a sub-balance already exists, BRM compares the new balance data with the following data in the existing valid sub-balances:

- Contributor
- Grantor object
- Rollover data
- Valid-from date
- Valid-to date

If the data matches, BRM adds the amount to the existing sub-balance; otherwise, it creates a new sub-balance.

For information about configuring sub-balances, see "About Configuring Sub-Balances".

How Resources in Validity-Based Sub-Balances Are Updated

By default, BRM stores common resources with the same validity periods in the same sub-balance, provided they are associated with the same balance group. (You can create separate balance groups per service in Pricing Center. See "Tracking Resources by Service".) BRM automatically creates a new sub-balance for resources with a unique validity period, if one does not already exist.

For example, an account owns two services that each include 100 free minutes that are always valid. The account has a balance of 200 free minutes stored in a single sub-balance. When the customer uses free minutes from each service, the free minutes are consumed from the common sub-balance.

Common resources with different validity periods are tracked in separate sub-balances. For example, an account owns two services that each include 100 free minutes. Free minutes for service 1 expire at the end of the month, and free minutes for service 2 expire at the end of the year. Each set of free minutes is stored in a separate sub-balance.

> **Note:**
>
> When common, noncurrency resources are configured to start on first usage, BRM creates a new sub-balance for each resource whether or not they have the same validity period. See "About Noncurrency Sub-Balances That Start on First Usage".

You can specify the order in which sub-balances are consumed by setting up resource consumption rules. See "Specifying the Order in Which Resource Sub-Balances Are Consumed".

You can also limit how validity-based resources such as free minutes are summed by configuring sub-balances. For example, you might want to limit usage of free minutes to a specific service or a specific call session. See "About Configuring Sub-Balances".

Configuring Time-Stamp Rounding for Validity Period Start Times

You can configure BRM to round time stamps to midnight for resources granted by cycle and purchase events.

See the following:

- Configuring Time-Stamp Rounding for Cycle Grants
- Configuring Time-Stamp Rounding for Purchase Grants

> **Note:**
>
> When the resource's purchase, cycle, and usage start and end units are set to 1 (seconds), 2 (minutes), or 3 (hours), and the validity period is less than 24 hours, time stamps are not rounded, regardless of your system configuration. If the validity is greater than 24 hours, the cycle, usage, and purchase end time stamps are rounded for the purpose of calculating the scale to determine the cycle fee amount to charge.

Configuring Time-Stamp Rounding for Cycle Grants

To configure BRM to round time stamps to midnight for the resources granted by cycle events:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **timestamp_rounding** entry to **1**.
3. Save and close the file.

Configuring Time-Stamp Rounding for Purchase Grants

To configure BRM to round time stamps to midnight for the resources granted by purchase events:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **timestamp_rounding** entry to **1**.
3. Save and close the file.
4. Go to the *BRM_Home*/**sys/data/config** directory, where *BRM_Home* is the directory in which BRM is installed.
5. Run the following command, which creates an editable XML file from the **rating** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

   This command creates an XML file named **bus_params_rating.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.
6. Open the **bus_params_rating.xml.out** file.
7. Search for the following line:

   ```
   <TimestampRoundingForPurchaseGrant>disabled</TimestampRoundingForPurchaseGrant>
   ```
8. Change **disabled** to **enabled**.
9. Save the file as **bus_params_rating.xml**.
10. Run the following command, which loads this change into the appropriate **/config/business_params** object.

    ```
    pin_bus_params PathToWorkingDirectory/bus_params_rating.xml
    ```

    where *PathToWorkingDirectory* is the directory in which **bus_params_rating.xml** resides.

    > **Caution:**
    >
    > BRM uses the XML in this file to overwrite the existing **rating** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of BRM's rating configuration.

    > **Note:**
    >
    > To run the command from a different directory, see the description for **pin_bus_params** in *BRM Developer's Guide*.
11. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
12. Stop and restart the CM.
13. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About Noncurrency Sub-Balances That Start on First Usage

When a noncurrency resource is configured to start on first usage (when the customer first consumes the resource balance), BRM always creates a new sub-balance for that resource when it is granted. A new sub-balance is created for each resource even when a product or discount grants multiple first-usage resources of the same type. (For more information about first-usage start times, see "About Balance Impacts That Become Valid on First Usage".)

For example, a product includes two grants of 100 free minutes. Each grant starts on first usage and ends at the end of the cycle. When the product is purchased, BRM adds two free-minute sub-balances to the account, one for each grant.

Keeping first-usage resources in separate sub-balances enables those balances to have unique validity periods, which are set individually when each resource is consumed. For example, if there are two balances of 100 free minutes each and the customer makes a 30-minute phone call, the validity period of only the balance that was consumed is set when the call is made.

A resource balance that has a first-usage start time will remain available for consumption for as long as the balance is not used. For example, on January 1, a customer purchases a deal that includes 30 free text messages that are valid for 30 days from first usage. The text message balance remains unused in the account until April 18, when the customer sends the first text message. The validity period of the text message balance is then set to start on April 18 and end 30 days later, on May 17.

About Configuring Sub-Balances

You can optionally configure sub-balances to track resources for more specific types of usage such as:

- Free minutes per call session.
- Frequent flyer miles per service instance.
- Friends and family calls to specific locations.
- Discount amounts shared with sponsored accounts.

You configure sub-balances for the entire BRM system by editing and loading a configuration file.

By default, common resources for all services owned by an account are placed in the same sub-balance if they have the same validity period and are associated with the same balance group. (You can create separate balance groups per service in Pricing Center. See "Tracking Resources by Service".)

To limit the allocation and consumption of resources, you configure sub-balances by specifying the following values:

- *Resource ID*. This is the ID for the type of resource in the sub-balance, such as dollars or free minutes.
- Event type. This is the type of event that impacts the sub-balance, such as GSM usage events (**/event/session/telco/gsm**).
- *Contributors*. Contributors can be any field in the event record or object. There are two types of contributors:

- *Retrieving contributor*. All sub-balances with common retrieving contributors are summed when sub-balances are retrieved. For example, to retrieve the total balance for specific services, specify the service object as the retrieving contributor.
- *Updating contributor*. A sub-balance is created or updated by balance impacts for each unique updating contributor. For example, to add or deduct free minutes for specific dialup sessions, specify the session object as the updating contributor.

About Sub-Balance Contributors

Sub-balance contributors are specified by a field name from the event record or object. The value in the field you specify is used to retrieve and update the sub-balances. Some fields you might want to use as contributors include:

- The service object.

  Specify a service field to track resources for specific service instances such as fax, telephony, and text messaging.

  > **Note:**
  >
  > Specifying the service object in the configuration is one way of creating service-level balances. The other way is to create a balance group for the service when you define your plans in Pricing Center. See "Tracking Resources by Service".

- The session object.

  Specify a session field to track resources per session instance. This is useful when you offer discounts for certain levels of usage (for example, 10 frequent flyer miles per hour of phone calls).

- The product object.

  When several products in the same plan have different rollover rules, tracking resources per product permits BRM to update the free minutes for each product.

- The phone number field.

  Specify a phone number field to track resources for called telephone numbers.

- The account object.

  Specify the account field to track resources that are shared among several accounts, such as earned free minutes. When free minutes are distributed, they can be divided based on each account's usage level.

Retrieving and Updating Sub-Balances

Sub-balances are retrieved for a given resource. For example, when CSRs use Customer Center to view an account's current cash balance, all valid cash sub-balances (sub-balances with current validity periods and matching contributors) are summed to provide the current resource balance.

Sub-balances are updated by a balance impact. When a balance impact affects more than one sub-balance, the sub-balances are updated in the following order:

1. **By validity period**. By default, sub-balances are updated in chronological order based on the validity start date or end date. You specify the order in which sub-balances are used by

setting up resource consumption rules. See "Specifying the Order in Which Resource Sub-Balances Are Consumed".

2. **By contributor**. Sub-balances with specific field contributors are impacted before sub-balances that accept all contributors (with an asterisk or empty contributor values). See "Configuring Sub-Balances".

Configuring Sub-Balances

To configure sub-balances, you edit the sub-balance configuration file and then run the **load_pin_sub_bal_contributor** utility to load the contents of the file into the **/config/sub_bal_contributor** object in the BRM database.

Editing the pin_sub_bal_contributor File

Each line in the **pin_sub_bal_contributor** file defines the usage for which a sub-balance is created. The configurations apply to all sub-balances in the BRM database, but they are implemented at the balance group level. That is, when a usage event occurs, if the account has more than one balance group, only the sub-balances within the specified balance groups are impacted.

To add sub-balance configurations, use this syntax:

```
resource_type:event_type:retrieving_contributor:updating_contributor
```

For example:

```
1000003:/event/session/gprs:PIN_FLD_SESSION_OBJ:PIN_FLD_SESSION_OBJ
```

where:

- **1000003** is the resource type for frequent flyer miles.
- **/event/session/gprs** is the event type that impacts the frequent flyer miles resource.
- **PIN_FLD_SESSION_OBJ** is the retrieving contributor. Therefore, a separate balance summary is retrieved for each unique GPRS session.
- **PIN_FLD_SESSION_OBJ** is the updating contributor. Therefore, separate sub-balances are created or updated for each unique GPRS session.

The contributors can be a specific field or an asterisk (*) to indicate any contributor. For example, the following entry retrieves and tracks dollars (resource type 840) in a single sub-balance for all events with any contributor:

```
840:/event:*:*
```

To configure several sub-balances for one resource type, leave subsequent resource fields empty. If no resource is specified, the previous resource specified is used. For example, the next two configurations use the same resource (100002):

```
100002 : /event/session/telco/gsm : PIN_FLD_SESSION_OBJ : PIN_FLD_SESSION_OBJ
       : /event/session/dialup : PIN_FLD_SERVICE_OBJ : PIN_FLD_SERVICE_OBJ
```

Running the load_pin_sub_bal_contributor Utility

> **Note:**
>
> When you run the **load_pin_sub_bal_contributor** utility, it replaces the existing sub-balance configurations. If you are updating a set of sub-balance configurations, you cannot load new configurations only. You load complete sets of sub-balance configurations each time you run the **load_pin_sub_bal_contributor** utility.

> **Note:**
>
> The resources specified in the **pin_sub_bal_contributor** file must exist in the BRM database before the **load_pin_sub_bal_contributor** utility is run. Therefore, you must first load the **pin_beid** file by running the **load_pin_beid** utility. See "load_pin_beid".

Use the following command to run the **load_pin_sub_bal_contributor** utility:

```
load_pin_sub_bal_contributor pin_sub_bal_contributor
```

If you are not in the same directory as the **pin_sub_bal_contributor** file, include the complete path to the file. For example:

```
load_pin_sub_bal_contributor BRM_Home/sys/data/pricing/example/pin_sub_bal_contributor
```

For more information, see "load_pin_sub_bal_contributor".

To verify that the sub-balance configurations loaded correctly, display the **/config/sub_bal_contributor** object by using Object Browser, or use the **robj** command with the **testnap** utility.

Sub-Balance Configuration Example

The following examples show how sub-balances can be configured for various business needs.

Sub-Balances per Service Example

There are two ways to track resources per service instance:

- By creating a balance group for the service. See "Tracking Resources by Service".
- By specifying the service object as a contributor in the **pin_sub_bal_contributor** file.

Table 15-1 shows contributor configurations to track resources for GSM services. The resources are dollars and free minutes.

Table 15-1 Contributor Configurations

Resource	Event Type	Retrieving Contributor	Updating Contributor
Dollars	/event/session/gsm	PIN_FLD_SERVICE_OBJ	PIN_FLD_SERVICE_OBJ
Free minutes	/event/session/gsm	PIN_FLD_SERVICE_OBJ	PIN_FLD_SERVICE_OBJ
Free minutes	/event/cycle_forward	PIN_FLD_SERVICE_OBJ	PIN_FLD_SERVICE_OBJ
Free minutes	/event/cycle_forward	PIN_FLD_SERVICE_OBJ	PIN_FLD_SERVICE_OBJ

Each line in the above example specifies a configuration.

- **Dollars** is a currency sub-balance that is created in the account balance group.
- **/event/session/gsm** is the event type that will impact the dollars resource.
- **PIN_FLD_SERVICE_OBJ** is the retrieving contributor, so the dollars resource for each unique GSM service is summed when retrieving balance information.

  If you specify an asterisk instead of PIN_FLD_SERVICE_OBJ for the retrieving contributor, dollars for *all* GSM services owned by the account would be summed when retrieving balance information.
- **PIN_FLD_SERVICE_OBJ** is the updating contributor, so a separate sub-balance is created to track dollars for every unique GSM service that the account owns, such as telephony, fax, and SMS.
- **Free minutes** is a noncurrency sub-balance that is created in the account balance group.
- **/event/session/gsm** is the event type that will impact the free minutes resource.
- **PIN_FLD_SERVICE_OBJ** is the retrieving contributor, so the free minutes resource for each unique GSM service is summed when retrieving balance information.

  If you specify an asterisk instead of PIN_FLD_SERVICE_OBJ for the retrieving contributor, free minutes for *all* GSM services owned by the account would be summed when retrieving balance information.
- **PIN_FLD_SERVICE_OBJ** is the updating contributor, so a separate sub-balance is created to track free minutes for every unique GSM service that includes free minutes. Additional free minutes and consumption of free minutes are restricted to the GSM service instance.

  If you specify an asterisk instead of PIN_FLD_SERVICE_OBJ as the updating contributor, free minutes granted by every GSM service owned by the account would be shared by all the GSM services.
- **Free minutes** is a noncurrency sub-balance that is created in the account balance group.
- **/event/cycle_forward** is the event type that will impact the free minutes resource.
- **PIN_FLD_SERVICE_OBJ** is the retrieving contributor, so free minutes granted by cycle forward events is summed for each unique service when retrieving balance information.
- **PIN_FLD_SERVICE_OBJ** is the updating contributor, so a separate sub-balance is created to track free minutes granted by cycle forward events for each unique service that grants free minutes. Consumption of free minutes granted at the beginning of the cycle is restricted to the service that granted them.

Specifying the Order in Which Resource Sub-Balances Are Consumed

Customers are sometimes granted multiple sub-balances of a particular resource, such as minutes. For example, a customer's balance might include minutes granted at the start of the accounting cycle and rollover minutes from the previous month. Because the minutes have different validity periods, they are grouped into different resource sub-balances. See "About Noncurrency Sub-Balances".

When the customer uses a service, BRM needs to know which minutes (or sub-balance) to use first. You use resource consumption rules to specify the order in which resource sub-balances are consumed, according to the validity start time and end time.

For example, to use rollover minutes first, you configure BRM to consume sub-balances based on the earliest validity start time. To use the minutes that expire first, you configure BRM to consume sub-balances based on the earliest validity end time.

Consumption rule descriptions

BRM supports the resource consumption rules shown in Table 15-2:

Table 15-2 Supported Consumption Rules

Consumption Rule	Description
Earliest start time (EST)	Consume the sub-balance with the earliest validity start time first.
Latest start time (LST)	Consume the sub-balance with the latest validity start time first.
Earliest expiration time (EET)	Consume the sub-balance with the earliest validity end time first.
Latest expiration time (LET)	Consume the sub-balance with the latest validity end time first.
Earliest start time and latest expiration time (ESTLET)	Consume the sub-balance with the earliest validity start time first. If multiple sub-balances have the same start time, use the one with the latest end time first.
Earliest start time and earliest expiration time (ESTEET)	Consume the sub-balance with the earliest validity start time first. If multiple sub-balances have the same start time, use the one with the earliest validity end time first.
Latest start time and earliest expiration time (LSTEET)	Consume the sub-balance with the latest validity start time first. If multiple sub-balances have the same validity start time, use the one with the earliest validity end time first.
Latest start time and latest expiration time (LSTLET)	Consume the sub-balance with the latest validity start time first. If multiple sub-balances have the same validity start time, use the one with the latest validity end time first.
Earliest expiration time and earliest start time (EETEST)	Consume the sub-balance with the earliest validity end time first. If multiple sub-balances have the same validity end time, use the one with the earliest validity start time first.
Earliest expiration time and latest start time (EETLST)	Consume the sub-balance with the earliest validity end time first. If multiple sub-balances have the same validity end time, use the one with the latest validity start time first.
Latest expiration time and earliest start time (LETEST)	Consume the sub-balance with the latest validity end time first. If multiple sub-balances have the same validity end time, use the one with the earliest validity start time first.

Table 15-2 (Cont.) Supported Consumption Rules

Consumption Rule	Description
Latest expiration time and latest start time (LETLST)	Consume the sub-balance with the latest validity end time first. If multiple sub-balances have the same validity end time, use the one with the latest validity start time first.

For example, if a customer's balance includes the following minutes and the customer makes a phone call on February 10, the consumption rules specify which group of minutes are impacted first:

- 100 Anytime Minutes with a validity period of February 1 to February 28.
- 50 rollover Anytime Minutes with a validity period of January 1 to February 28.
- 200 bonus Anytime Minutes with a validity period of January 15 to June 15.

If the consumption rule is set to earliest start time (EST), BRM applies the balance impact to the 50 rollover Anytime Minutes first. Likewise, if the rule is set to earliest expiration time and latest start time (EETLST), BRM applies the balance impact to the 100 Anytime Minutes first.

You specify the order in which resource sub-balances are consumed in your price plans. You can also set systemwide and default resource settings. BRM reads and uses the consumption rule settings in the order shown below:

1. **Loan consumption rules**. If a loan sub-balance is present, it will be consumed first. If there are multiple loan sub-balances, they are consumed according to the existing consumption rules.
2. **Price plan setting**. Your price plans can include resource-to-consumption rule mappings for each service that you support. This enables you to have different consumption rules for the same resource based on the plans owned by the customer. When a customer purchases a plan, the plan's resource-to-consumption rule mapping is stored in the customer's **/balance_group** object. If there are any conflicting rules for the same resource, BRM uses the rule from the most recently purchased plan.
3. **Systemwide resource setting**. You can specify a systemwide resource-to-consumption rule mapping for each service that you support. BRM uses the systemwide settings only if a rule is not defined for the resource in the customer's purchased plans. BRM stores systemwide settings in the **/config/beid** object.
4. **Default resource setting**. You can specify a default setting that applies to all resources. This setting is used only if a consumption rule is not defined for the resource in the customer's purchased plans or if there is not a systemwide resource setting. BRM stores the default resource setting in the **/config/business_params** object.

How BRM Applies Consumption Rules

To apply resource consumption rules, BRM automatically orders an account's resource sub-balances whenever the account adds or changes a resource. For example, if you modify a resource with a consumption rule of EET, BRM orders the resource sub-balances based on the validity end time, starting with the earliest expiration time first. When the account has a balance impact, BRM searches through the account sub-balances, in order, and applies the balance impact to the first sub-balance that matches the following criteria:

- Has a validity period that matches the event's time stamp.
- Has a balance to consume.

If there is any remaining balance impact, BRM applies it to the next sub-balance that matches the criteria. This process continues until all of the sub-balances have been depleted. BRM then restarts the search from the beginning and applies any remaining balance impact to the first sub-balance that matches the validity period.

For example, assume that an account with the following resource sub-balances makes a 30-minute phone call on June 4. If the consumption rule is set to LSTEET, BRM first consumes the 5 minutes from sub-balance A and then consumes the 10 minutes from sub-balance C. Because the account's sub-balances have been depleted, BRM charges the remaining 15 minutes to sub-balance A.

- Sub-balance A has a balance of 5 minutes with a validity period of June 1 to June 15.
- Sub-balance B has a balance of 0 minutes with a validity period of June 1 to June 30.
- Sub-balance C has a balance of 10 minutes with a validity period of May 1 to July 15.
- Sub-balance D has a balance of 0 minutes with a validity period of January 1 to December 30.

The method BRM uses to implement resource consumption rules is different for batch rating and real-time rating.

How Batch Rating Applies Consumption Rules

In batch rating, resource consumption rules are applied by the DAT_BalanceBatch module. The DAT_BalanceBatch module orders resource sub-balances when Pipeline Manager starts and when an account adds or modifies a resource. When processing CDRs, FCT_ApplyBalance uses the DAT_BalanceBatch module to search through an account's resource sub-balances, in order, and find the first sub-balance that matches the CDR time stamp and has a balance to consume.

By default, Pipeline Manager supports all of your price plan, systemwide, and default consumption rule settings. However, you can configure Pipeline Manager to use the default setting only by using the **UseFlexibleConsumptionRule** registry entry in the DAT_BalanceBatch module:

- **True**: Uses the consumption rules defined for each resource in a balance group. If a consumption rule is not defined, it uses the rules defined in the **/config/beid** object. If a consumption rule is not defined in a balance group or in the **/config/beid** object, the module uses the rule defined in the **multi_bal** instance of the **/config/business_params** object.
- **False**: Uses the systemwide consumption rule defined in the **multi_bal** instance of the **/config/business_params** object only.

How Real-Time Rating Applies Consumption Rules

In real-time rating, consumption rules are applied by the Balance FM opcodes. When an account is created or modified, the Balance FM opcodes read the consumption rule for each resource and order the resource sub-balances appropriately. When a customer has a balance impact, the Balance FM opcodes search through the customer's resource sub-balances, in order, and apply the balance impact to the first sub-balance that matches the event time stamp and has a balance to consume.

Setting Resource Consumption Rules

To assign resource consumption rules, perform one or more of these tasks:

- Setting Consumption Rules in Your Price Plans
- Setting Systemwide Consumption Rules for Each Resource
- Setting the Default Consumption Rule

For information about changing your consumption rule settings, see "Modifying Resource Consumption Rules".

Setting Consumption Rules in Your Price Plans

You set resource consumption rules in your price plans by using Pricing Center. See the Pricing Center Help for more information.

> **Note:**
>
> You can also use the "loadpricelist" utility to assign resource consumption rules in your price plans. You specify the rules in the **price_list.xml** file's **consumption_rule** field and then load the file into the database by using the **loadpricelist** utility. See "Using the XML Pricing Interface to Create a Price List ".

Pricing Center and **loadpricelist** store your price lists and consumption rule settings in **/plan** objects in the BRM database. When a customer purchases a product, the plan's resource-to-consumption rule mapping is stored in the customer's **/balance_group** object.

If you use a custom client application to create price lists, you must modify your application to pass consumption rule settings to the Pricing FM opcodes. For more information, see "Customizing Credit Limits and Resource Consumption Rules" in *BRM Opcode Guide*.

Setting Systemwide Consumption Rules for Each Resource

You define systemwide consumption rules for each resource that you support by using Resource Editor. See the Resource Editor Help for more information.

> **Note:**
>
> You can also use the **pin_beid** configuration file to assign systemwide resource consumption rules. You specify the rules in the file's **con_rule_id** column and then load the file into the database by using the **load_pin_beid** utility. See "load_pin_beid" for more information.

Resource Editor and **load_pin_beid** store the systemwide consumption rule setting for each resource in the **/config/beid** object.

Setting the Default Consumption Rule

The default consumption rule applies to all resources in your system. The default setting is earliest start time and earliest expiration time (ESTEET). You can change this setting by modifying a field in the **multi_bal** instance of the **/config/business_params** object created during BRM installation.

You modify the **/config/business_params** object by using the **pin_bus_params** utility.

To set the default resource consumption rule:

1. Use the following command to create an editable XML file for the **multi_bal** class:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

   This command creates the XML file named **bus_params_multi_bal.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

   ```
   <SortValidityBy>ESTEET</SortValidityBy>
   ```

3. Change **ESTEET** to the appropriate value:

   - Earliest start time (EST)
   - Latest start time (LST)
   - Earliest expiration time (EET)
   - Latest expiration time (LET)
   - Earliest start time and latest expiration time (ESTLET)
   - Earliest start time and earliest expiration time (ESTEET)
   - Latest start time and earliest expiration time (LSTEET)
   - Latest start time and latest expiration time (LSTLET)
   - Earliest expiration time and earliest start time (EETEST)
   - Earliest expiration time and latest start time (EETLST)
   - Latest expiration time and earliest start time (LETEST)
   - Latest expiration time and latest start time (LETLST)

   For a description of each setting, see "Consumption rule descriptions".

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **multi_bal** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

4. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_multi_bal.xml
   ```

   You should run this command from the *BRM_Home*/**sys/data/config** directory, which includes support files used by the utility.

5. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

6. Stop and restart the Connection Manager (CM).

7. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Modifying Resource Consumption Rules

You can modify the resource consumption rules that you set in your price plans, in your **/config/beid** object, and in your **/config/business_params** object at any time. However, the new and changed settings apply to new customers only. Your existing customers will continue to use the previous settings:

- Changes to plan-level consumption rules do not trigger any update to existing customers that own the changed plan. Existing plan owners continue to use the plan's previous settings.
- Changes to the **/config/beid** object do not trigger changes to existing customers. Existing customers continue to use the previous systemwide settings.
- Changes to the **/config/business_params** object do not trigger changes to existing customers. Existing customers continue to use the previous default setting.

> **Note:**
>
> Pipeline Manager cannot use the new or modified **/config/beid** or **/config/business_params** consumption rule settings until they are loaded into pipeline memory. See "Reloading Data into a Pipeline Manager Module".

About Rollovers

Use the rollover feature to specify the quantity of unused resources that can be rolled over into subsequent cycles.

For example, you can configure a product so that a portion of unused free minutes from each cycle can be rolled over for use in the following cycle or cycles.

> **Note:**
>
> Rollover resources are distinct from resources issued through one-time grants, such as those issued by a CSR or a discount product. One-time grants are valid between two specific dates.

You set up rollovers in a product's event map in Pricing Center.

By default, rollover events are included in the **pin_event_map** file.

About Rollover Resource Sub-Balances

BRM maintains each rollover resource as a sub-balance. Each sub-balance specifies a resource balance that is valid (available for use) between valid-from and valid-to dates stored in BRM. The total amount of a resource available to a customer for each cycle is equal to the sum of all resource sub-balances that are valid during the cycle.

BRM validates whether a sub-balance can be rolled over by checking the rules that govern rollover (such as the amount to roll over, the maximum rollover amount allowed, the maximum number of cycles to roll over, and so on).

When a rollover event occurs, one of three things happens to the balance in a currently valid resource sub-balance:

- **If the full amount is eligible for rollover**, BRM does one of the following:
  - Creates a new rollover sub-balance for the rolled over amount. The rollover sub-balance validity period has the same valid-from date as the original sub-balance, and its valid-to date is extended to the end of the new cycle.
  - Adds the rolled over amount to an existing sub-balance if the existing sub-balance has the same data as would the new rollover sub-balance (such as the resource type, rollover rules, valid-from and valid-to dates, sub-balance contributors, and so on).
- **If only a portion of the resources is eligible for rollover**, the amount in the sub-balance is divided into a non-rollover sub-balance and a rollover sub-balance. The non-rollover sub-balance has the same valid-to date as the original sub-balance. Its resource balance is used for late-arriving usage events. The valid-to date for the rollover sub-balance is extended to the end of the new cycle. Its resources are available for the customer to use in the new cycle.
- **If none of the resource is eligible for rollover**, the sub-balance is not changed. This condition occurs when the sub-balance has already been rolled over the maximum number of times allowed. Amounts in this sub-balance are available only for late-arriving usage events.

> **Note:**
>
> Your system's memory limits might limit the number of months you can roll over a sub-balance. The number of resource sub-balances that BRM must maintain varies according to the number of rollover cycles allowed. You set the maximum allowed rollover cycles when configuring the rollover balance impact in Pricing Center.

CSRs can view rollover resource balances by using Customer Center.

When Rollover Events Occur

Resource sub-balances can be rolled over to another cycle as soon as they surpass the valid-to date (expire). Most resource sub-balances expire on the account's billing day of month (DOM). However, some sub-balances may expire in the middle of a cycle due to a product or service cancellation or due to flexible cycles.

Because many BRM features depend on sub-balances being rolled over the day after they expire, BRM rolls over resources at the following times:

- **At the end of the billing cycle**: BRM automatically rolls over all eligible resource sub-balances to the next cycle as part of the billing process. This catches all resource sub-balances that expire on the account's billing DOM.
- **When you run the pin_bill_day script**: The **pin_bill_day** script automatically runs the **pin_rollover** utility as part of the billing process. The **pin_rollover** utility rolls over any resource sub-balances that have expired but have not been rolled over to the next cycle. This catches any resource sub-balances that expired in the middle of the cycle.

Deleting Expired Sub-Balances

To delete expired sub-balances, use the **pin_sub_balance_cleanup** utility.

Rollover Example

The following example demonstrates:

- Granting resources (free minutes) at the start of each cycle (calendar month).
- Rolling over unused free minutes from previous months at the start of each month.

> **Note:**
>
> For information about rolling over resources that expire in the middle of a cycle, see "About Rolling Over Resources That Expire in Midcycle".

- Dividing an initial resource grant into rollover and non-rollover sub-balances.
- Limiting the total number of free minutes that can be rolled over from previous months.
- Dividing a rollover balance into rollover and non-rollover sub-balances when the Maximum Cumulative Rollover Total value is reached.
- The default order in which free minutes are consumed from the valid resource sub-balances.
- Expiration of free minutes in a sub-balance when the sub-balance has been rolled over the number of cycles specified by the Maximum Number of Rollover Cycles value.

In this example, the customer's product includes:

- 500 free minutes granted at the beginning of each usage cycle.
- A rollover specifying that:
  - Up to 100 unused free minutes from each month can be rolled over.
  - The maximum number of rollover cycles is 2.
  - A limit of 150 *total* rollover minutes from previous months can be rolled over into a new month.

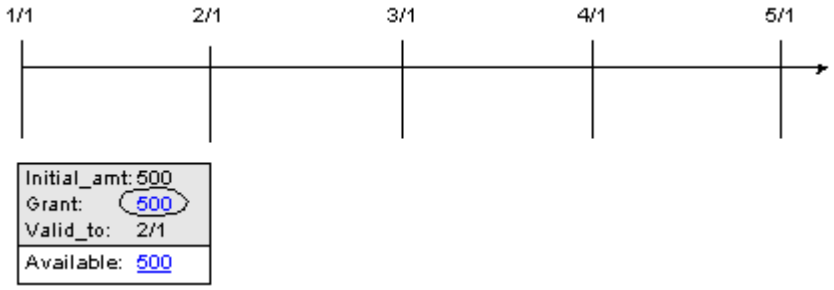
Other assumptions:

- Cycles are monthly and start on the first day of each calendar month.
- The product with the rollover is valid starting January 1.
- The customer consumes no free minutes until March.

> **Note:**
>
> In this example:
>
> - New sub-balances, or those that have a change in their validity period, are highlighted in gray.
> - New grant resource values or changes to current resource amounts within the sub-balances are in **bold black**.
> - Components of available resource totals are in **blue**.
> - Components of used resource totals are in **red**.

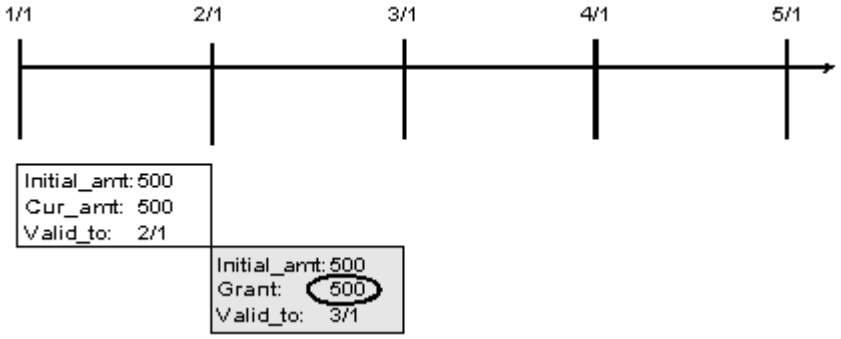
1. On January 1, the customer is granted 500 free minutes for use during January as shown in Figure 15-1. The free minutes are maintained in a resource sub-balance.

Figure 15-1 Initial 500 Free Minutes Grant in January



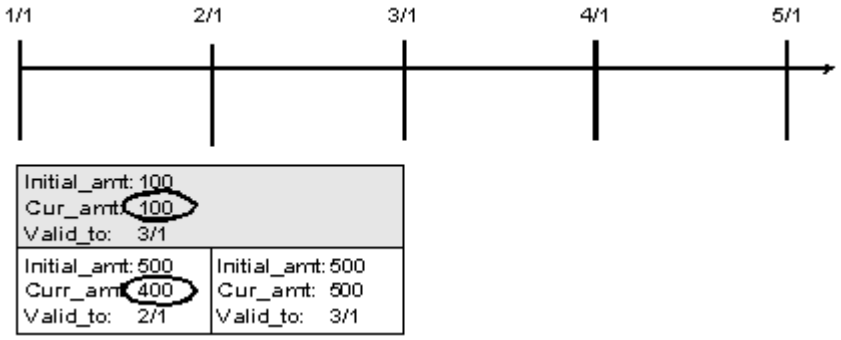
2. On February 1:
   - The cycle forward event creates a new resource sub-balance to track the grant of 500 free minutes for use during February as shown in Figure 15-2.

Figure 15-2 February Free Minutes Grant



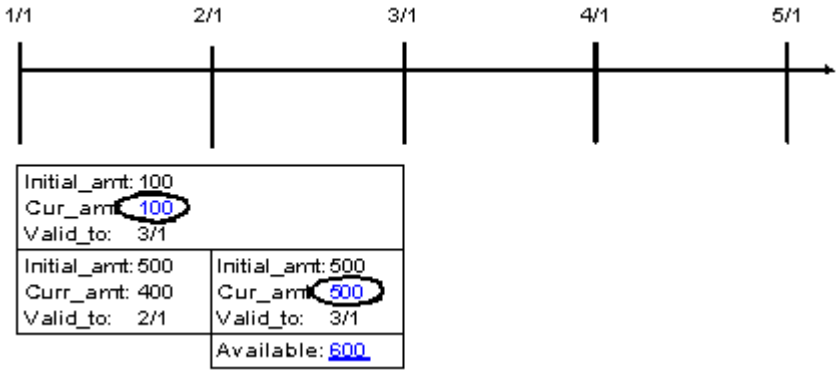
   - The cycle rollover event creates a new resource sub-balance for tracking unused January free minutes that roll over into February. 100 minutes are put in the new sub-balance, and they are valid until March 1. The original sub-balance for January is decremented 100 minutes, leaving 400 minutes available for late-arriving January events as shown in Figure 15-3.

Figure 15-3 January Free Minutes Rollover



The customer now has 600 free minutes (500 granted February 1 plus 100 rolled over from January) available for use during February as shown in Figure 15-4.

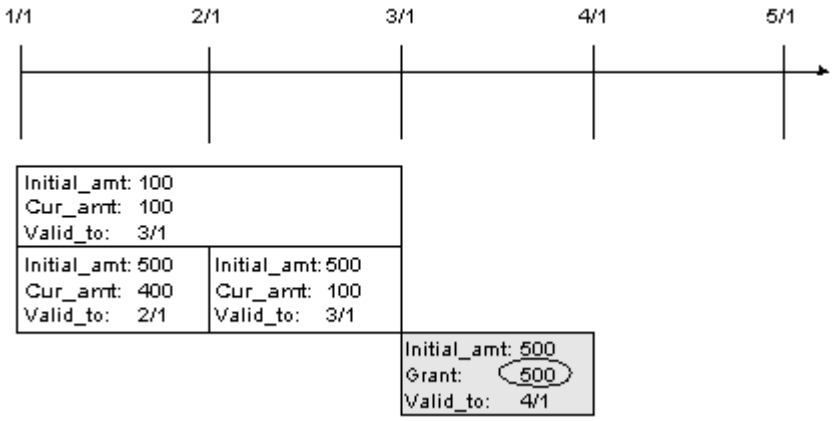
Figure 15-4 February Total Free Minutes Available



3. On March 1:

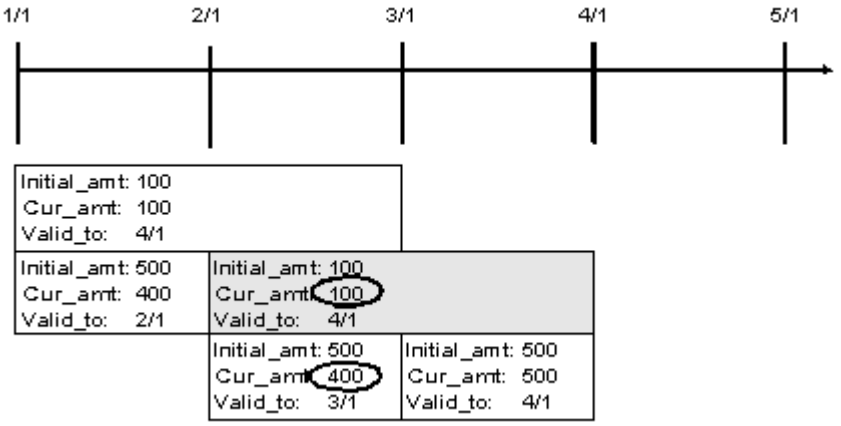
   - The cycle forward event creates a new resource sub-balance to track the grant of 500 free minutes for use during March as shown in Figure 15-5.

Figure 15-5 March Free Minutes Grant



   - The cycle rollover event creates a new resource sub-balance for tracking unused February free minutes that roll over into March as shown in Figure 15-6. 100 minutes are put in the new sub-balance, and they are valid until April 1. The original sub-balance for February is decremented by 100 minutes, leaving 400 minutes available for late-arriving February events.

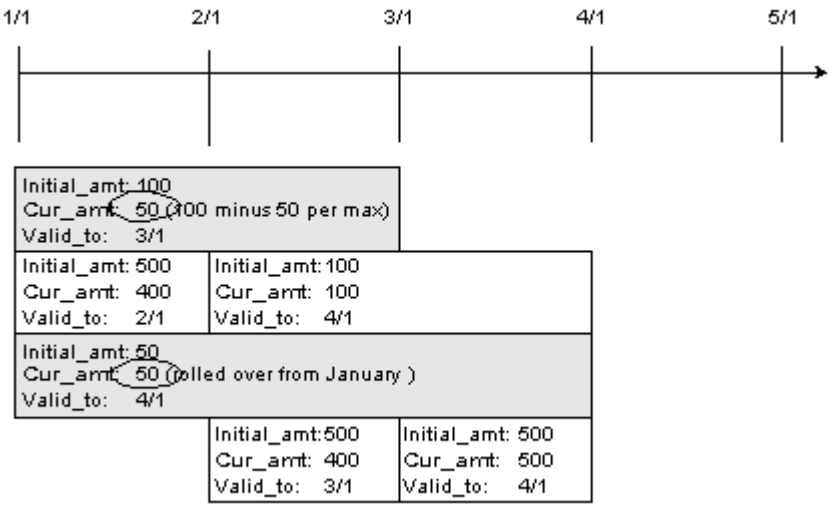
Figure 15-6 February Free Minutes Rollover



- The cycle rollover event divides the January rollover minutes into two sub-balances to enforce the rule that only 150 *total* rollover minutes for a resource can carry forward into a new month. Because 100 free February minutes are already rolled over, only 50 minutes can be rolled over from the January rollover sub-balance for use in March.

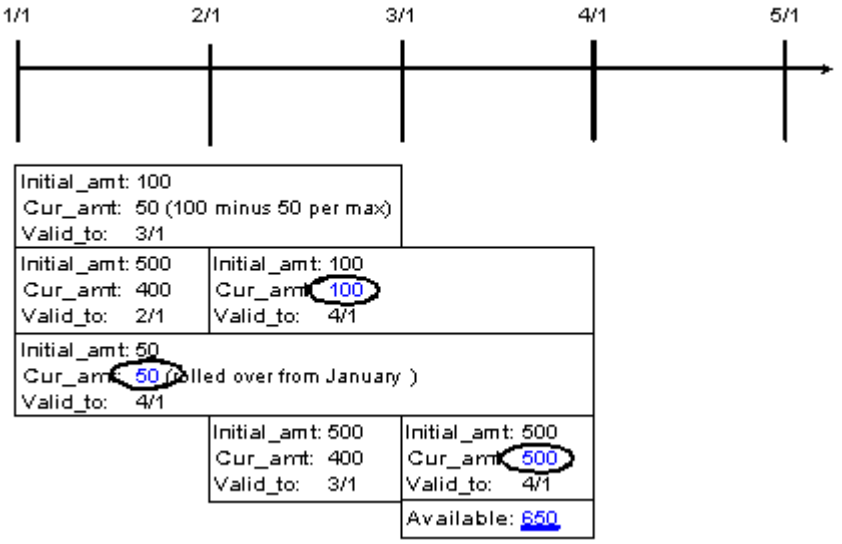
  One new sub-balance contains 50 minutes available for use for late-arriving February and January events. The other contains 50 rollover minutes that the customer can use in March. The resource balances for each month are shown in Figure 15-7.

Figure 15-7 Remaining January Free Minutes Rollover



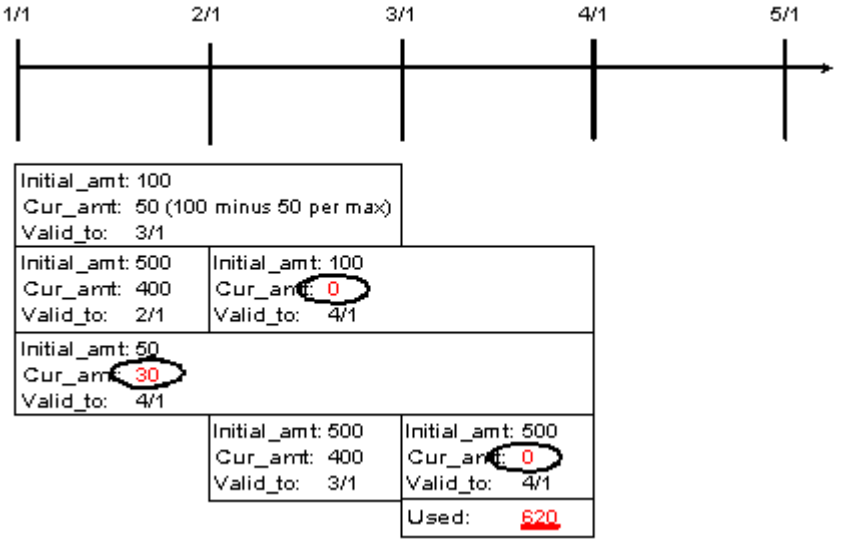
The customer now has 650 free minutes (500 granted March 1 plus 100 rolled over from February plus 50 rolled over from January) available for use during March as shown in Figure 15-8.

Figure 15-8 March Total Free Minutes Available



4. During March, the customer consumes 620 minutes as shown in Figure 15-9. Sub-balance resources are used starting with the newest sub-balance, as indicated by the sub-balance valid-from dates:
   - All 500 free minutes from the March grant are consumed, leaving a zero balance.
   - All 100 February rollover minutes are consumed, leaving a zero balance.
   - 20 of the 50 January rollover minutes are consumed, leaving 30.

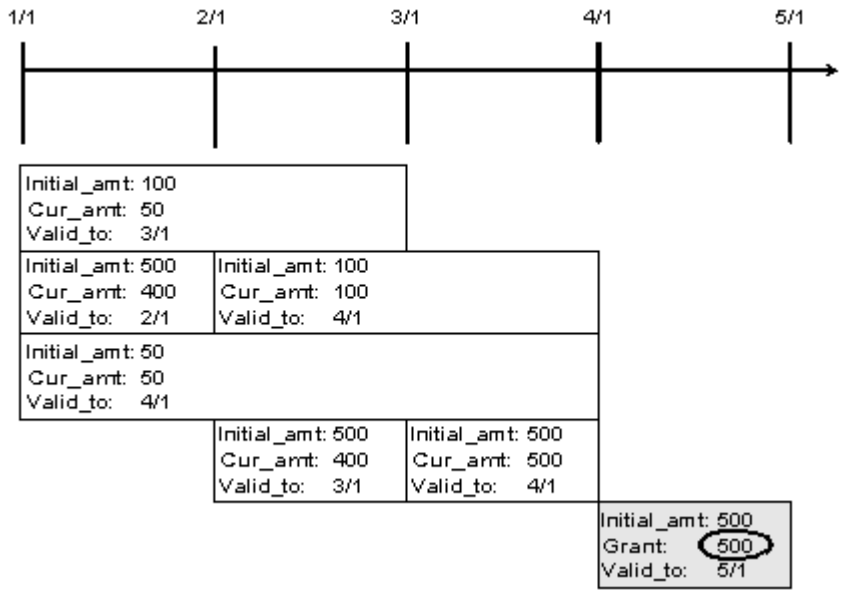
Figure 15-9 March Free Minutes Usage



5. On April 1, the cycle forward event creates a new resource sub-balance to track the grant of 500 free minutes for use during April as shown in Figure 15-10.

   No free minutes roll over into April from the March and February rollovers because these resources were consumed in March. The 30 free minutes remaining from the January rollover are not rolled over into April because the maximum number of cycles a grant can be rolled over is set to 2.

Figure 15-10 April Free Minutes Grant



The customer has only 500 free minutes available for April.

> **Note:**
>
> - Rollover sub-balances are not truncated or prorated when the corresponding product is canceled.
> - If two products contributing to the same balance group have rollover rateplans configured for the same resource with the same rollover frequency, either of the products can be used to roll over the resources. In this case, rollover results may vary depending on the product that is selected first.

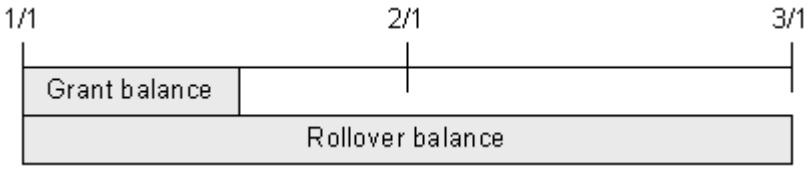
About Rolling Over Resources That Expire in Midcycle

A resource balance can expire in the middle of a cycle: for example, when the resource is valid for only minutes, hours, or days or when the resource is valid for one or more months and starts in the middle of a month.

If a resource's validity period does not end when the cycle ends, the rollover sub-balance is valid for the entire cycle in which the resource is granted and for the whole of the next cycle.

In Figure 15-11, a monthly cycle event grants 60 free minutes that are valid for two weeks from the grant date. The free minutes are granted on January 1. On January 14, when the balance of free minutes expires, any remaining resources that can be rolled over are added to a new sub-balance, which is valid from January 1 to March 1:

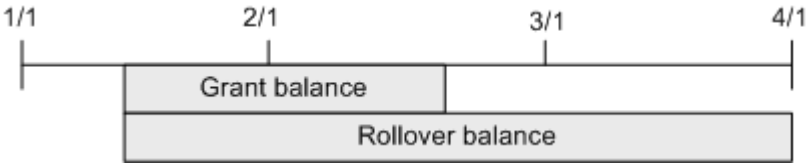
Figure 15-11 Grant and Rollover Balance Date Validity Example 1



The same rollover period applies when the validity period of the free minutes ends after the cycle in which they are granted. In Figure 15-12, a monthly cycle event grants 300 free minutes that are valid for six week from the grant date. The free minutes are granted on January 1. On February 11, when the balance expires, any remaining resources that can be rolled over are added to a new sub-balance, which is valid from January 1 to March 1:

If the resource's validity period does not start at the beginning of the last cycle, the resources are rolled over for one entire cycle.

Figure 15-12 Grant and Rollover Balance Date Validity Example 2



Prorating Rollover Resources

Customers typically purchase and cancel products at some point in the middle of a cycle. When you set up a rollover in Pricing Center, you can specify whether rollover resources are prorated for the first and last cycles based on the number of days in the cycles that the product is owned.

Table 15-3 describes the rollover proration options:

Table 15-3 Rollover Proration Options

Proration Option	Description
Rollover entire amount	Roll over the available monthly rollover resources.
No rollover	Do not roll over any available monthly rollover resources.
Prorate rollover amount	Calculate the rollover resources based on the percentage of the cycle that the customer owned the rollover.

If you choose to prorate the rollover amount, BRM uses the equation in Figure 15-13 to calculate the amount to rollover:

Figure 15-13 Calculation for Amount to Rollover

$$\frac{ValidFrom - ValidTo}{DaysInCycle} \times Rollover$$

where:

- *ValidFrom* is the validity period's starting date. For example, if the validity period is from March 15 to April 14, ValidFrom is March 15.
- *ValidTo* is the validity period's ending date. For example, if the validity period is from March 15 to April 14, ValidTo is April 14.
- *DaysInCycle* is the number of days in the current cycle. For example, if the validity period is from March 15 to April 14, DaysInCycle is 31.
- *Rollover* is the available monthly rollover resource, such as 100 minutes.

For example, suppose:

- A customer buys a product on January 15 that grants 500 free minutes of use during each cycle.
- A rollover is set up so that up to 200 unused free minutes roll over into the following cycle.
- The usage cycle is a calendar month.
- The customer does not use any minutes during the first cycle.

Table 15-4 describes how the rollover is handled, depending on the Purchase Mid-cycle Proration setting:

Table 15-4 Impact of Proration Options

Proration Option	Amount Rolled Over into February
Rollover entire amount	200 free minutes are rolled over from January into February. In February, the customer has 700 free minutes (200 rolled over plus the 500 free minutes February grant).
No rollover	No free minutes are rolled over from January into February. In February, the customer has only the February grant of 500 free minutes.
Prorate rollover amount	Approximately 110 free minutes are rolled over from January into February: (17/31) * 200 = 109.67 In February, the customer has about 610 free minutes (110 rolled over plus the 500 free minutes February grant).

A similar set of results applies to the last-month proration calculation specified in the Cancel Mid-cycle Proration setting.

About Rolling Over Free Resources during Plan Transition

When your customers transition from one plan to another, you can specify that free noncurrency resources be rolled over from one plan to another if both plans belong together to at least one plan list.

For more information on plan transitions, see "Transitioning between Plans".

To specify free resources rollover between plans during plan transition, you must perform these tasks:

- If you use a custom application for customer management, provide the trigger for controlled rollover in the input flist to the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode.

- When you create your plan lists, ensure that plans between which you want to allow rollovers are in at least one plan list together.

  For more information, see "About Plan Lists".

These general rules apply to controlled rollovers:

- Controlled rollovers are not affected by the product cycle rollover settings.

  For more information on cycle rollovers, see "About Rollovers".
- A controlled rollover does not count as a rollover and hence is not restricted to the maximum rollover quantity and the rollover units per period settings for the cycle rollover.
- The cycle grants of free resources that have been prorated during the plan transition are rolled over to the next plan.

16

About Real-Time Rating

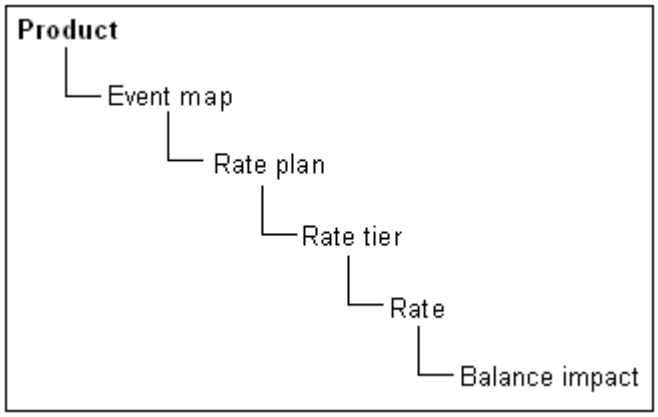
This chapter describes how to perform Oracle Communications Billing and Revenue Management (BRM) real-time rating if you are using Pricing Center for creating price lists.

Topics in this document:

About Organizing Real-Time Rates in a Product

You organize rates in a product in a hierarchical structure as shown in Figure 16-1:

Figure 16-1 Product Rate Structure



This structure enables you to create multiple components at each level; for example, you can rate multiple events in a single product, create multiple rates based on dates, and create multiple balance impacts for a rate based on previously rated quantities.

About Real-Time Rate Plans

Usually, you use one rate plan per event. A rate plan includes the following:

- The rate or rates that apply to the event
- The currency to use for the event

  By default, rate plans use the system currency.
- Tax information, such as when to calculate the tax, and the tax code

  Assigning tax codes to rate plans enables you calculate taxes more accurately.

- For cycle forward fees, billing in advance

  See "Charging Cycle Forward Fees in Advance".

You can use multiple rate plans to choose a rate plan based on event data (for example, telephone call origins and destinations). See "Real-Time Rating Using Event Attributes".

In rare cases, you might need to use custom event analysis code to specify how to rate an event. See "About Custom Event Analysis".

You use rate plans to define how much to charge for an event. Rate plans include a currency, a tax code, and at least one rate tier. See "About Organizing Real-Time Rates in a Product".

> **Note:**
>
> - Rate plan names and descriptions (for example, the names of rate plans, rate tiers, and date ranges) can include a maximum of 255 characters.
> - If you use pipeline batch rating to rate events, you must use the same rate plan names in your real-time rate plans and in the pipeline rate plans.

Specifying the Rate Plan Currency

You specify the currency in which each customer's fees are charged. See "Managing system and account currencies" in *BRM Managing Customers*.

Specifying the Rate Plan Tax Code

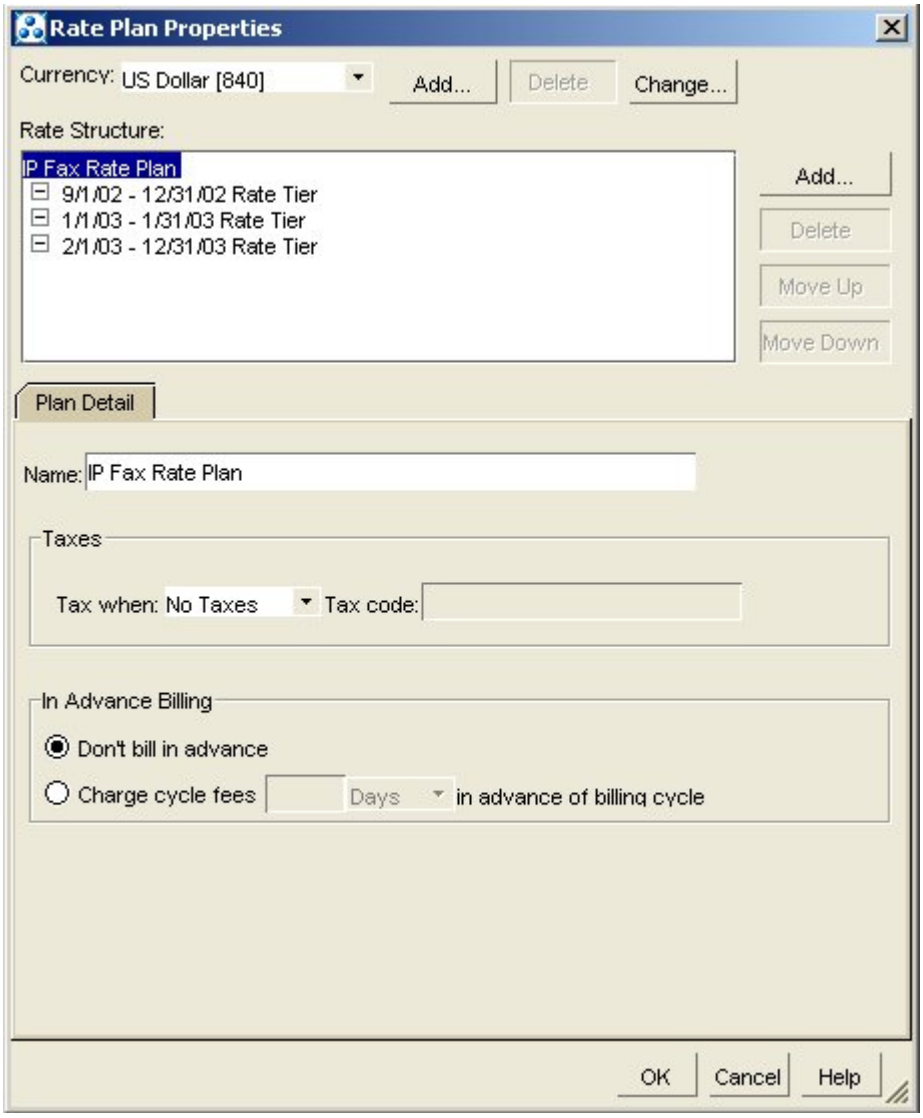
You can assign a tax code to enable BRM or a third-party tax calculation package to calculate taxes. See "Creating Tax Codes" in *BRM Calculating Taxes*.

About Rate Tiers

You use rate tiers to define when and for how long rates are valid and which rates are applied first.

Figure 16-2 shows a rate plan that includes three rate tiers. Each of the rate tiers is valid for a range of dates:

Figure 16-2 Rate Plan with Three Rate Tiers



For each rate tier, you can specify one of the following:

- **Absolute date range**. This rate tier is in effect only in the dates specified (for example, March through September).
- **Relative date range**. This rate tier is in effect only in a range of days relative to the product purchase (for example, for 30 days after the purchase date).

> **Note:**
>
> For either absolute or relative date ranges, you can specify that a rate tier's valid period starts immediately and never ends. In effect, this makes the rate tier always valid.

You need more than one rate tier only in the following cases:

- To apply a balance impact for different resources in a specific order. For example, if you give customers free hours, you can prioritize rate tiers to charge for hours before currency.
- If you create rates based on date or time. For more information about rate tiers, see "Real-Time Rating Based on Date and Time".

About Real-Time Rates and Balance Impacts

A rate defines the balance impact that results when the event is rated (for example, $2 for a one-hour dialup session). For information about balance impacts, see "Specifying How an Event Affects an Account Balance".

You usually use only one balance impact per rate, but you can specify multiple balance impacts for a single rate. For example:

- $1 per fax
- 10 points toward free faxes

In addition, you can group balance impacts in terms of quantity. For example:

- For the first 10 faxes:
  - $1 per fax
  - 10 points toward free faxes
- For the next 90 faxes:
  - 50 cents per fax
  - 25 points toward free faxes
- For 100 or more faxes:
  - 5 cents per fax
  - 50 points toward free faxes

For more information, see "Real-Time Rating Based on Event or Resource Quantity".

Specifying How an Event Affects an Account Balance

You specify the impact each event has on an account balance. You define each balance impact in terms of resources, general ledger IDs, proration, sponsorship, and discounts. In addition, you can assign an impact category to each balance impact.

The balance impact is the affect that rating has on an account balance. A balance impact can be positive or negative:

- A positive, or *credit*, impact is a charge to the customer (for example, $2 for an hour of Internet usage). The customer owes you the amount of the credit balance impact.
- A negative, or *debit*, impact is a resource given to the customer (for example, 10 free Internet access hours per month). You owe the customer the amount of the debit balance impact.

About Scaled Impacts

An event that uses a *scaled* resource amount impacts an account balance immediately when the event starts and accumulates until the event ends. For example, BRM starts rating an IP telephony call as soon as it starts. This enables BRM to track the impact to the account as the event is rated.

A scaled amount affects a resource balance by multiplying the scaled amount value by the event quantity. For example:

Scaled rate of $1 per hour **X** event quantity of 10 hours **=** balance impact of $10

Use scaled impact when a resource is calculated per month, per hour, or by customer usage:

- $1 per faxed page
- $19.95 per month for interactive games service
- 10 free hours per month for Internet service
- $.50 per email sent by the customer
- Purchase rates for products that customers can purchase in quantity

  For example, you might sell a product that provides 100 hours of service a month. If customers can purchase two of these products for a total of 200 hours, use the scaled impact for the purchase rate.

As a rule, when you do not know the exact amount that will be rated, specify scaled impact.

Even though monthly rates are consistent from one month to the next, you still use a scaled impact. This is because they charge an amount per event; in this case, a monthly event generated by BRM:

Scaled rate of $10 per month **X** event quantity of 1 month **=** balance impact of $10

About Fixed Balance Impacts

A *fixed amount* is a predefined amount that is always applied to the account when any amount of the event quantity is rated.

Use a fixed impact to define rates based on occurrence. These are amounts that do not change with usage or with the quantity of events being rated. For example:

- $5 installation fee
- $10 cancellation fee
- 10 hours of free service when you register for a service

An event that uses a fixed amount does not impact the account balance until the event ends. This enables BRM to calculate variable fees for the same event type. For example, when a single balance impact has both a fixed amount and a scaled amount, BRM can sum the impact before the account balance is affected.

Specifying Multiple Balance Impacts for a Single Rate

You can specify more than one balance impact per rate:

- For the same event, you can apply a balance impact to a currency resource and to a noncurrency resource. For example, you might create a product that charges $20 per month and gives the customer 10 free Internet access hours. In that case, the same rate would have two balance impacts: one for US dollars and one for Internet hours.
- For the same event, you can apply a balance impact based on quantity. For example:
  - For 0 to 10 faxes, charge $0.50 per fax
  - For 11 or more faxes, charge $0.10 per fax

  For more information, see "Real-Time Rating Based on Event or Resource Quantity".

Specifying the Validity Period of Granted Resources

You specify validity periods for balance impacts that grant resources such as free minutes. The validity period defines when a granted resource is available for consumption by the subscriber's usage. The subscriber can consume the resource balance during the valid period only.

You can set the balance impact to start immediately, relative to when the resource is granted, or on first usage (when the subscriber consumes the resource for the first time). Setting resources to start on first usage is useful, for example, to provide free resources for a limited time, beginning when the subscriber starts consuming the resource. For more information, see "About Balance Impacts That Become Valid on First Usage".

Assigning Impact Categories to Balance Impacts

You use impact categories to apply different balance impacts when using the same rate plan. For example, to rate calls made to two different countries, you create impact categories for each country. When the call event occurs, the impact category determines which balance impacts to apply.

If you are using a rate plan selector, you assign impact categories to balance impacts when you create a rate. When the rate plan selector selects a rate plan and impact category, only those balance impacts with the matching impact category are applied. See "Real-Time Rating Using Event Attributes".

Assigning General Ledger (G/L) IDs to Balance Impacts

To collect information for your general ledger accounting system, you associate types of balance impacts with general ledger (G/L) IDs.

For example, when you create a usage rate, you can associate it with a G/L ID that records the revenue from all usage-rate balance impacts.

Allowing Charges to Be Sponsored

When charges are sponsored by a sponsor group, the sponsor group owner account receives the sponsored balance impacts generated by the group's member accounts. Therefore, you can use sponsor groups to let one customer pay part or all of other customers' service charges.

You define which accounts are sponsor group owners and which are members by creating sponsor groups in Customer Center. When you create a sponsor group, you select rate plans for the group to sponsor. You can select rate plans from all the products in your company's price list except the default product.

> **Note:**
>
> Creating sponsor groups is easier if all resources in the rates in sponsored rate plans are sponsorable. You might want to name your products so they can be identified as products that contain sponsorable rate plans.

> **Note:**
>
> If your company supports branded service management, this list displays all the products in your company's price list that are available for the sponsor group owner's brand. If your company does not support brands, this list displays all the products in the price list.

One Sponsor per Product

Each products in a member account should have only one sponsor. If group A sponsors purchase charges for a service and group B sponsors monthly subscription charges for the same service, the purchase and cycle rate plans should be in different products. If both rate plans are in the same product, BRM assigns all the product's charges to only *one* of the two sponsor groups. If the charges are assigned to group A, only the purchase fees for the service are sponsored. Members of group B must pay their own subscription fees.

You can also provide commissions by using service provider management. See "About Remittance".

About Balance Impacts That Become Valid on First Usage

You can set the validity period of resources granted by products and discounts to start on first usage: when subscribers begin consuming the resource balances for the first time.

For example, a deal grants 300 free minutes and 5 days of free video streaming. You set the free minutes to be valid immediately and set the streaming video to be valid when first used. A subscriber purchases the deal on June 1 and uses the video streaming service for the first time on June 5. The free minutes are valid starting June 1 and the streaming video is valid from June 5 to June 10.

You can also set the products and discounts that grant resources to start on first usage. There is no dependency between a product or discount's first usage start time and the first usage start time of the resource it grants. For information about products and discounts that start on first usage, see "About Effective Periods That Start on First Usage".

Resources whose validity period starts on first usage are added to the account balance at the time of the grant, but the validity period is not set until one of the following occurs:

- (Default) The first usage session in which the resource is consumed ends. The validity start time is set to the end time of the session.
- The resource is first authorized to be consumed. The validity start time is set to the start time of the first session.

The end time is set based on the end time that you configure for the resource and is set relative to the start time.

For more information, see "About Setting Resource Validity Periods Based on First Usage".

> **Note:**
>
> If the product granting the resource also starts on first usage, when the product is first used and its validity period is set, resources that are granted by that product cannot be consumed by the first usage event during real-time rating. This is because real-time rating does not generate the purchase and cycle events that grant resources until after the first usage event is rated. However, resources granted by products are available for consumption during real-time discounting because discounting occurs after the purchase and cycle events have been processed. For information about products and discounts that start on first usage, see "About Effective Periods That Start on First Usage".

If a product or discount is canceled after being used, the validity end time of any resources granted by the product or discount is set to the time of the cancellation.

About Setting Resource Validity Periods Based on First Usage

> **Note:**
>
> This section does not apply to products and discounts whose validity periods start on first usage. For information about them, see "About Effective Periods That Start on First Usage".

To set a resource validity period based on first usage, BRM must first set the period's start time. The end time is then set relative to the start time.

A resource validity period's start time is set during one of the following phases of the first usage session:

- **Stop Accounting:** (Default) The period's start time is set to the end time of the first usage session. This is recommended for typical validity periods.
- **Authorization:** The period's start time is set to the start time of the first usage session. This is recommended for short validity periods, such as one-day promotions. It prevents users from extending the promotional period by leaving the usage session open and thus preventing BRM from setting the validity period's end time.

  > **Note:**
  >
  > If usage does not occur after the validity period's start time is set during the authorization phase (for example, a phone call is authorized but not answered), the validity period's start time nonetheless remains in effect and cannot be reset.

To specify when the start time is set, use the **SetFirstUsageInSession** business parameter. See "Setting Validity Start Time to the Start Time of the First Usage Session".

Setting Validity Start Time to the Start Time of the First Usage Session

By default, start times for resource validity periods based on first usage are set to the end time of the first usage session. Alternatively, they can be set to the start time of the session. You

change the time they are set to by using the **pin_bus_params** utility. For information about that utility, see "pin_bus_params" in *BRM Developer's Guide*.

To set validity start time to the start time of the first usage session:

1. Go to the *BRM_Home*/**sys/data/config** directory, where *BRM_Home* is the directory in which BRM is installed.
2. Run the following command, which creates an editable XML file from the **activity** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsActivity bus_params_act.xml
   ```

   This command creates the **bus_params_act.xml.out** file in your working directory. To place this file in a different directory, include the path in the file name.
3. Open the **bus_params_act.xml.out** file.
4. Search for the following line:

   ```
   <SetFirstUsageInSession>disabled</SetFirstUsageInSession>
   ```

5. Change **disabled** to **enabled**.
6. Save the file as **bus_params_act.xml**.
7. Go to the *BRM_Home*/**sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_act.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_act.xml** resides.

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing **act** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, those changes affect the associated aspects of the BRM activity configuration.

9. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
10. Stop and restart the Connection Manager (CM).
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About First-Usage Start Time for Shared Resources

If a resource is shared among accounts in a resource sharing group, the resource validity period is set when any account in the group first impacts the resource balance. Because the same balance is shared with all accounts in the group, the validity period of that resource applies to all accounts.

About Synchronizing First-Usage Validity of Resources in Deals

You can synchronize the validity periods of all resources granted in a deal that have first-usage start times. This sets the validity of all granted resources with first-usage start times when one of those granted resources is first consumed.

For example, a deal grants free minutes and free text messages and both are set to start on first usage. A subscriber purchases the deal and on June 5 uses the free minutes by making a phone call. When the call is rated, the validity periods for both free minutes and text messages are set to start on June 5.

Only first-usage resources in the deal that is used to rate the event are synchronized. The validity periods of other resources and first-usage resources granted by other deals are not changed.

If a resource with a first-usage start time is granted after the validity period of other first-usage resources has already been set, the validity of the newly granted resource is synchronized with the existing validity periods.

To synchronize first-usage resources, you select **Align deal resource validity on first usage** in the **Deal Attributes** tab of Pricing Center.

About Restricting Resource Validity End Time to the Product or Discount End Time

For resources that start on first usage, you can restrict the resource validity end times so they do not extend beyond the validity end time of the products or discounts that grant the resources. This ensures that the resource balance cannot continue to be consumed after the product or discount expires.

> **Note:**
>
> When a product or discount is *canceled*, the validity period end time of resources granted by that product or discount is set to the time of the cancellation.

About Fold Events

A fold event is a special internal event that usually occurs at the end of the accounting cycle. A fold event can be configured to either change a resource balance (currency or noncurrency) to zero or convert the resource balance into other resources.

For example, you can create a fold event to cancel unused free hours at the end of each month. You can also create a fold event to credit frequent flyer miles for every $500 a customer spends on a service.

Fold events are triggered at the following times:

- During regular billing
- At product cancellation
- When Bill Now is invoked

When a fold event occurs, BRM checks whether the resource sub-balance or bucket is valid or expired. The resource is folded if the sub-balance is expired or is always valid.

For example:

- If 500 free minutes are valid forever; the resource can be folded at any time.
- If 500 free minutes expire on February 1, it can be folded after January 31 23:59:59. The end time of the fold event must be January 31 23:59:59 or later.

> **Note:**
>
> If you use delayed billing, folds occur at the end of the delayed billing period. This is to ensure that delayed events can continue to consume granted resources before final billing is run. However, free resources are granted at the beginning of each cycle, before the fold event occurs. If the newly granted resources do not have a configured validity period (that is, their validity period never ends), the fold event will remove those resources.
>
> When you use fold events to remove unused resources and you also use delayed billing, you should always configure a validity period for the granted resources. This prevents fold events from removing resources that are granted at the beginning of a cycle before final billing is run.

You can configure fold events to convert resource R1 to R2. You can also convert R2 to R3 if resource R2 is available or tracked in the account's balance group *before* the fold event is triggered.

For example, a price plan has a cycle forward event that charges a $100 cycle fee and grants 1000 free minutes. When you create an account with this plan, the account's balance group contains two resources: USD and Free Minutes. If you configure a fold event to:

1. Convert the free minutes (R1) to free games (R2)
2. Convert the free games (R2) to Euro (R3)
3. Convert the free minutes (R1) balance to zero

The free games resource (R2) is not converted to R3 in step 2 because it is not available in the account's balance group when the fold is triggered, even though Free Games is added to the account's balance group in step 1.

To fold free games to Euro, you add Free Games to the plan in the **Track Balances** tab in the Plan Attributes window. When the account is created, Free Games is added to the account's balance group and its balance is set to zero (0).

By default, an account's resource balances are folded in ascending order based on the resource ID. You can change this order by modifying the policy opcode, PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD.

You can create a fold to charge for overuse of free resources before the customer cancels a product.

Ensuring That All of an Event Is Rated

BRM cannot apply a rate if the credit limit for its resource has been reached. However, there might be times when an event, or part of an event, has not been rated, and all credit limits have been reached. To charge for the unrated event or event portion, you create a *limit override rate*. A limit override rate is used even though a credit limit has been reached.

In effect, a limit override rate sets the baseline charge for an event. When no other rates apply because of credit limits, you charge a default amount for the event.

17

About Pipeline Rating

This chapter provides an overview of how to use the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager to rate usage events.

Topics in this document:

About Batch Rating

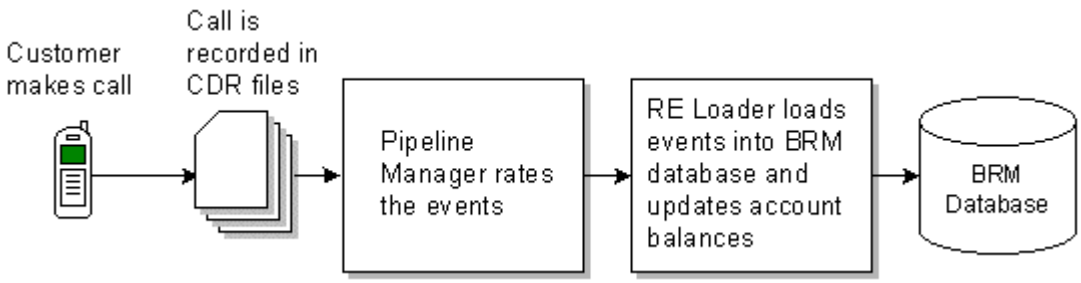
Batch rating is typically used for rating telephony services, where large amounts of events are recorded in files. Batch rating is usually performed by Pipeline Manager, but you can also perform batch rating by using Universal Event (UE) Loader.

For example, to rate telephone calls:

1. Pipeline Manager reads the data from CDRs to determine the time and duration of the calls.
2. Pipeline Manager rating modules apply a charge to the amount of time. For example, if you specify a charge of 10 cents per minute, a 10-minute call costs $1. This charge is called the *balance impact*.
3. Rated Event (RE) Loader loads the rated events into the BRM database.
4. BRM adds the total charge for the event to the customer's account balance.

Figure 17-1 shows how a billable event is rated by Pipeline Manager and recorded in the BRM database:

Figure 17-1 Pipeline Manager Event Rating and Recording to BRM Database



How Events Are Rated by Using Pipeline Manager

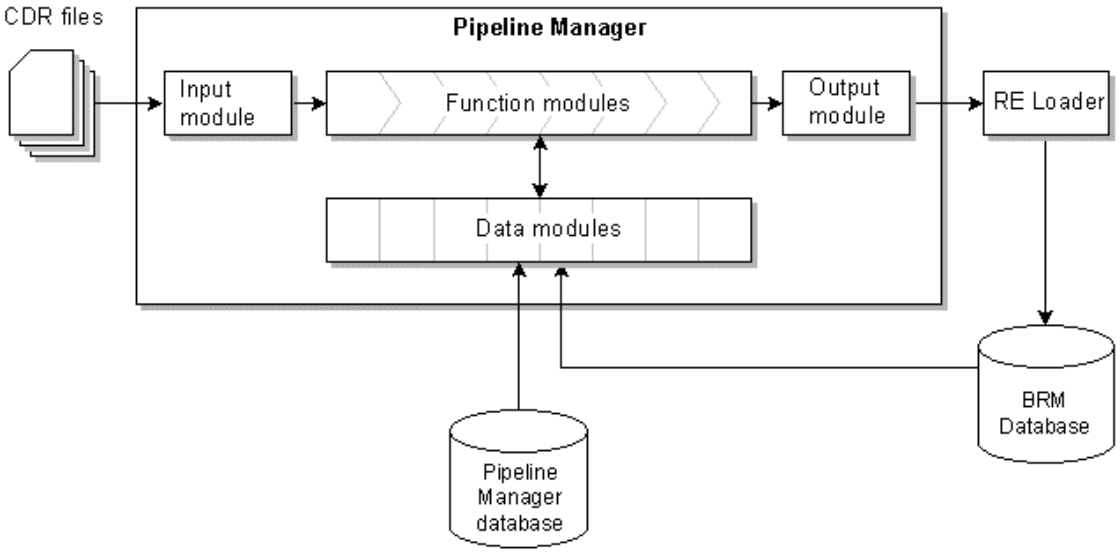
Pipeline Manager rates only usage events, not recurring events, and not one-time events. Pipeline Manager rates usage events as follows:

1. Call detail record (CDR) files are collected from network switches and processed by mediation software. The mediation software converts data into the Oracle CDR format so it can be read by a pipeline input module. See "About the Oracle CDR Format".

   Mediation can also remove records that are not needed for rating. For example, it can remove records for free text messages reminding customers of a missed call.
2. The mediation software places the CDR file in a predefined directory.
3. The pipeline input module reads the CDR file and begins processing it. The input module does the following:
   - Performs error checking on the input CDR.
   - Converts the data in the CDR file into event data records (EDRs). To do so, the input module normalizes the raw data that represents each event and formats it into a standard structure that can be processed by the pipeline modules.
4. Function modules, working with data modules, rate the events.
   - *Function modules* perform the preprocessing and rating operations. For example, they check for duplicated EDRs, determine the quantity to rate, the zone to apply, and the charge for the event.
   - *Data modules* supply data to the function modules by reading from the Pipeline Manager database or the BRM database.
5. The output module writes data to an output file.
6. Rated Event (RE) Loader reads the output file, loads the events into the BRM database, and updates the customer's account balance.

Figure 17-2 shows how usage events are rated:

Figure 17-2 Usage Events Rating



When you configure Pipeline Manager, you do the following:

- Configure the input module to accept and process your type of CDR file.
- Configure function and data modules to carry out normalization, zone mapping, and rating.
- Configure the output module to output the data.

In addition to configuring Pipeline Manager, you need to do the following:

- Configure the Account Synchronization Data Manager (DM) to transfer data from the BRM database to Pipeline Manager.
- Configure RE Loader to load rated events into the BRM database. See "Understanding Rated Event Loader".

How an EDR Is Processed in a Pipeline

When an EDR is sent through a pipeline, function modules perform the following types of operations:

- The input module processes the CDR and creates an event data record (EDR) for each event. Processing the CDR file includes:
  - Identifying each type of record in the file; for example, header records, event records, and trailer records.
  - Normalizing data and translating it into the internal EDR format.
- Preprocessing modules prepare EDRs for rating. For example:
  - The FCT_DuplicateCheck module discards duplicate EDRs.
  - The FCT_CallAssembling module assembles calls that have been split into multiple records.
  - The FCT_Reject module rejects EDRs with errors.

  Preprocessing modules typically run at the beginning of a pipeline.
- Enrichment modules normalize or add data that the rating modules need. For example:

  - The FCT_ServiceCodeMap module assigns an internal service code to identify which service generated the event.
  - The FCT_CliMapping module maps multiple phone numbers to a single phone number, so customers can be billed for all of their phones on one bill.

  Enrichment modules typically run before rating modules in a pipeline.
- Zoning modules calculate geographic or area-code-based zones for rating purposes.
- Rating modules perform rating.
- Discount modules perform discounting, after the EDR has been rated.
- The aggregation module collects data for reports.

About the Order of Modules in a Pipeline

Some modules need to be configured in a specific order. For example:

- You use the FCT_Discard module to discard EDRs that you don't want to rate. You need to discard them before the rating modules process them; otherwise you spend system resources on rating unwanted EDRs.
- To provide discounts, the discount module needs to work with events that have already been processed by the rating modules.

For information about the order of modules in a pipeline, see "Function Module Dependencies".

About the Oracle CDR Format

The Oracle CDR format is the standard CDR file format used by Pipeline Manager for processing CDRs. For example, the Oracle CDR format is used by pipelines to generate CDRs that are passed between pipelines for additional processing, such as between a preprocessing pipeline and a rating pipeline.

The Oracle CDR format can also be used as the input format for rating pipelines. Additionally, pipelines can be configured to translate other formats into the Oracle CDR format for rating.

For complete details about the Oracle CDR format structure, see "BRM Rating EDR Container Description".

For information about the input and output processes, see "Configuring EDR Input Processing" and "Configuring EDR Output Processing".

About EDRs

The data for each event is stored as an EDR. As an EDR is processed, function modules process data in it or add data. For example, the FCT_CustomerRating module adds the charge code. The FCT_MainRating module uses that code to calculate and add the charge amount.

The following sample shows a portion of a charge packet in an EDR. Each field stores a specific piece of data; for example, the RATEPLAN_CODE field stores the charge used for rating the event.

```
CHARGE_PACKET
RATEPLAN_CODE:              <SIMPLERATE>
RATEPLAN_TYPE:              <R>
ZONEMODEL_CODE:             <MT_ZM>
SERVICE_CODE_USED:          <SMS>
```

```
SERVICE_CLASS_USED:          <DEF>
IMPACT_CATEGORY:             <MT_SMS>
TIMEMODEL_CODE:              <TM_MTEL>
TIMEZONE_CODE:               <TZ_WKNDDT>
DAY_CODE:                    <WEEKEND>
TIME_INTERVAL_CODE:          <0800_2000>
PRICEMODEL_CODE:             <MT_SMS>
PRICEMODEL_TYPE:             <S>
RESOURCE:                    <YEN>
RUMGROUP:                    <EVENT>
RUM:                         <EVT>
CHARGE_TYPE:                 <N>
CHARGING_START_TIMESTAMP:    <20020901182200>
CHARGEABLE_QUANTITY_VALUE:   <1>
CHARGED_CURRENCY_TYPE:       <R>
CHARGED_AMOUNT_VALUE:        <50>
CHARGED_AMOUNT_CURRENCY:     <JPY>
CHARGED_TAX_TREATMENT:       <N>
CHARGED_TAX_RATE:            <0>
CHARGED_TAX_CODE:            <NORM>
USAGE_GL_ACCOUNT_CODE:       <50000>
```

About EDR Containers

EDR containers are temporary in-memory data structures for transporting EDRs through function modules. Each container stores data for a header record, a detail record, or a trailer record.

When a transaction starts:

1. The EDR Factory creates an EDR container according to the information in the container description file.
2. The input module writes data to relevant fields in the container. For more information, see "Configuring EDR Input Processing".
3. The function modules process data in or add data to particular fields. For example, the FCT_BillingRecord function module processes balance impact-related data.
4. The EDR Factory empties the container and releases the cache, which will be used for other containers.

About the EDR Container Description

The data in an EDR, the default values, and the way the data is organized are defined in a *container description*. A typical container description defines the following types of containers:

- **A header container type.** The header container type contains information stored in a header record; for example, the country of origin, originating network, and creation time. It also includes sequence numbers for ensuring that EDRs are processed correctly. A separate EDR is created for the header record.
- **One or more basic detail container types.** A basic detail container includes the data in an event record; for example, a phone call. This record includes information such as the A number and the service that generated the event. An EDR is created for each detail record. Each EDR can contain data for one service only; for example, GSM or GPRS.

  Each detail record includes one or more associated records. These records include service-specific data, zoning data, and rating data. See "About Associated Records".

- **A trailer container type.** The trailer container type contains information stored in a trailer record; for example, the number of records. A separate EDR is created for the trailer record.

In almost all cases, you can use the BRM EDR container description. You can also customize the data that is included in an EDR by customizing the container description.

> **Note:**
>
> If you customize the container description, you need to make sure that your customizations do not affect existing module functionality. For example, many modules require data from a specific EDR field.

About the Container Description File

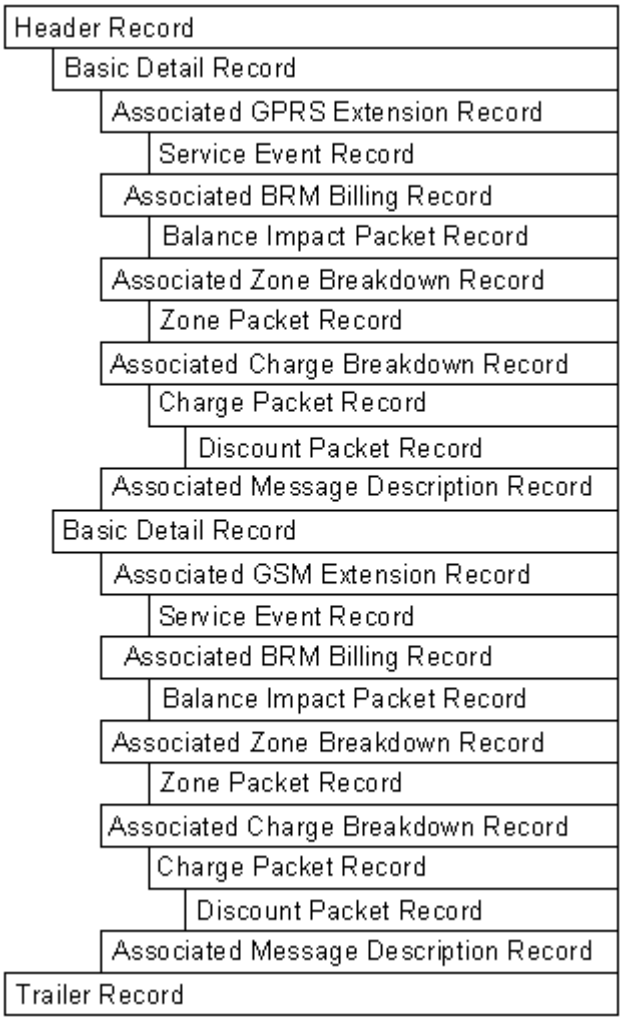
The container description file is an ASCII file that defines how to build EDR containers. You use the file to define the data in the EDR container, the default values, and the way the data is organized. For more information, see "About the EDR Container Description".

The default container description file is *pipeline_home*/**formatDesc/Portal/containerDesc.dsc**, where *pipeline_home* is the directory in which you installed Pipeline Manager. This file covers the needs of most input file formats, but you can customize it to meet your business requirements.

> **Note:**
>
> If you alter or add fields to the container description file, you may also need to write custom iScripts to access these new fields.

Figure 17-3 shows the BRM EDR container organization:

Figure 17-3 BRM EDR Container Organization



About Associated Records

Associated records are added to EDRs by different modules; for example, the FCT_SegZoneNoCust module adds an associated zone breakdown record.

- **Associated service extension record**. This record stores specific information about the service that generated the event; for example, the originating switch and the device number.

  EDRs for GSM events can include one or more service event records. This record includes event information, such as equipment used and details about supplementary services.

- **Associated BRM billing record**. This record stores rated event data that is loaded into the BRM database. It includes the POID of the **/account** object and **/service** object and the POID of the item that receives the balance impact. This record is created by the FCT_BillingRecord module.

  If an event affects more than one customer balance, an associated BRM billing record is created for each balance.

An associated BRM billing record can contain one or more *balance impact packets*. These contain data about the event. Each balance impact packet includes data for one balance impact per balance element, and, optionally, per G/L ID.

- **Associated charge breakdown record**. These records hold the charges for an event. For example, the FCT_CustomerRating module adds an associated charge breakdown record to the EDR to record the rating results.

  Each associated charge breakdown record includes one or more charge packets. Each charge packet includes a single charge; for example, the charge for a telephone call for a single time period.

  Each charge packet can include one or more *discount packets*. A discount packet includes information about the discount owner and rollover information.

  During rating, Pipeline Manager might generate several charge packets. For example:

  – When rating multiple balance elements, each balance element has its own charge packet.

  – When using multiple ratable usage metrics (RUMs), each RUM has its own charge packet.

  – When splitting charges across time zones, each time zone has its own charge packet.

  – If multiple currencies are used, each currency has its own charge packet.

- **Associated zone breakdown record**. This record contains data for zoning. It is created by the FCT_SegZoneNoCust module. This is used for comparative analysis of different zoning options.

  An associated zone breakdown record includes one or more *zone packet records*. Each of these records includes data about a single zone model.

- **Associated message description record**. This record holds information and error messages.

How an Input File Is Represented in EDR Containers

A typical conversion from an input file to EDR containers creates the following EDRs:

- A set of control EDRs that the pipeline uses for managing transactions. For example, this control EDR specifies to start a transaction:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <SERVICE>
Content type   = <BEGIN_TRANSACTION>
Originator     = <ifw.Pipelines.W_SAMPLE.Input>
Stream Number  = <0>
IsValidDetail  = <false>
Record number  = <1>
has Errors     = <0>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- A header record that includes information about the input file:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <DATA>
Content type   = <HEADER>
Originator     = <>
Stream Number  = <3>
IsValidDetail  = <false>
```

```
Record number  = <0>
has Errors     = <0>

HEADER
------

RECORD_LENGTH:                    <0>
RECORD_TYPE:                      <>
RECORD_NUMBER:                    <0>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- Multiple detail records, each containing the data for one event:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <DATA>
Content type   = <DETAIL>
Originator     = <>
Stream Number  = <3>
IsValidDetail  = <true>
Record number  = <1>
has Errors     = <0>

DETAIL
------

RECORD_LENGTH:                    <0>
RECORD_TYPE:                      <020>
RECORD_NUMBER:                    <0>
DISCARDING:                       <0>
TYPE_OF_A_IDENTIFICATION:         <S>
A_MODIFICATION_INDICATOR:         <00>
A_TYPE_OF_NUMBER:                 <0>
A_NUMBERING_PLAN:                 <0>
A_NUMBER:                         <008190115551212>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- A trailer record, which includes information about the records in the file:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <DATA>
Content type   = <TRAILER>
Originator     = <>
Stream Number  = <3>
IsValidDetail  = <false>
Record number  = <7>
has Errors     = <0>

TRAILER
-------

RECORD_LENGTH:                    <0>
RECORD_TYPE:                      <090>
RECORD_NUMBER:                    <14>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

- A set of control EDRs. This EDR specifies the end of a transaction:

```
= = = = =  S T A R T   O F   C O N T A I N E R  = = = = =
Container type = <SERVICE>
Content type   = <END_TRANSACTION>
Originator     = <ifw.Pipelines.W_SAMPLE.Input>
Stream Number  = <0>
IsValidDetail  = <false>
Record number  = <8>
has Errors     = <0>
.
.
.
= = = = = =  E N D   O F   C O N T A I N E R  = = = = = =
```

How EDRs Are Used for Managing Transactions

Transactions are managed by using the EDR content type:

- When an EDR with the BEGIN_TRANSACTION content type is processed, a new transaction is started. This usually occurs at the beginning of each input file.
- When an EDR with the END_TRANSACTION content type is processed, the transaction is ended. This usually occurs at the end of each input file.
- When an EDR with the STOP content type is processed, this indicates that the pipeline is shutting down gracefully and gives the modules a chance to save their state and prepare for the shutdown.

About Mapping EDR Field Names and Alias Names

The EDR container description defines the data in the EDRs and the name of each field. See "About the EDR Container Description".

In the pipeline modules, EDR fields can optionally be represented using two names:

- The EDR container field name; for example, DETAIL.RECORD_LENGTH. This is the input format of the field.
- An alias name, to which the container field name is mapped; for example, BDR_RECORD_LENGTH.

The sample below shows EDR container field names on the left and internal alias names on the right:

```
DETAIL.RECORD_LENGTH     ->  BDR_RECORD_LENGTH
DETAIL.RECORD_TYPE       ->  BDR_RECORD_TYPE
DETAIL.RECORD_NUMBER     ->  BDR_RECORD_NUMBER
DETAIL.DISCARDING        ->  DISCARDING
DETAIL.CHAIN_REFERENCE   ->  CHAIN_REFERENCE
```

Pipeline Manager function modules and iScripts use the alias name for manipulating data. When you write a custom iRule or an iScript, you must create an alias name for the EDR container fields that your module uses. Using alias names facilitates customization because you can use the container fields with different names and hierarchies without the need to change the module source code. See "Viewing and Creating Alias Mapping for an EDR Field".

> **Note:**
>
> The same EDR container field can be mapped to different alias names in different modules.

The list of BRM-defined alias names is in *pipeline_home***/formatDesc/Portal/AliasFieldList.dsc**.

Viewing and Creating Alias Mapping for an EDR Field

Use Pricing Center or Pipeline Configuration Center (PCC) to view a list of existing alias mappings and to add custom alias mappings.

The Alias Mapping table lists the existing alias names including the BRM-defined alias names. Each entry includes the following mapping information:

- **EDR Container Description**, which specifies the EDR container to which the EDR field belongs.
- **Reference**, which is the pipeline module that uses the alias name. Its value is the pipeline module reference in the module block of the pipeline registry file; for example, **PipelineSplit** in the following registry entry:

```
PipelineSplit
{
   ModuleName = FCT_IRules
   Module
  {
    …
   }
}
```

> **Note:**
>
> If the value of **Reference** is Account_CustA, Account_CustB, UniData_CustA, or UniData_CustB, the alias mapping table entry defines the EDR field from which to get the account identifier. If the value is RUM, the alias mapping defines the EDR field from which to get the rateable usage metrics (RUM) quantity. If the value is UOM, the alias mapping defines the EDR field from which to get the unit of measure for a RUM

- **Key**, which is the alias name for the EDR container field.

> **Note:**
>
> For Account_CustA, Account_CustB, UniData_CustA, and UniData_CustB, this is the service code.

- **Type**, which is **Internal** for the alias mappings used by the default pipeline modules and **Plugin** for the modules used by custom modules.
- **Field ID**, which is the name of the field in the EDR container.

- **History**, which specifies when the alias mapping was created or modified.

About Function Modules

Function modules perform all the rating tasks in a pipeline. In addition, they perform EDR management tasks, such as ensuring that duplicate EDRs aren't processed.

There are different categories of function modules. They generally run in the following order:

- Preprocessing modules prepare EDRs for rating. See "About Preprocessing Modules".
- Enrichment modules add data to EDRs. See "About Enrichment Modules".
- Service mapping modules map external service codes to internal service codes. See "About Service Mapping Modules".
- Zoning modules calculate zone data to use for rating. See "About Zoning Modules".
- Rating modules rate the events. See "About Rating Modules".
- Discount modules adjust the charges. See "About Discounting Modules".
- Roaming modules are specialized zoning and rating modules that handle roaming events. See "About Roaming Modules".

About Preprocessing Modules

Use the preprocessing modules to handle the following tasks:

- Use the "FCT_CallAssembling" module to assemble calls that have been split into multiple EDRs. See "Assembling EDRs".
- Use the "FCT_Reject", "FCT_PreSuspense", and "FCT_Suspense" modules to handle EDRs that contain errors. Rejected EDRs are sent to a separate output stream. You can then fix the problem that caused them to be rejected and re-process them. See "Suspending and Recycling EDRs".
- Use the "FCT_DuplicateCheck" module to discard duplicate EDRs. See "Handling Duplicate EDRs".
- Use the "FCT_Discard" module to discard EDRs that you don't want to process. See "Discarding and Skipping EDRs ".
- Use the "FCT_EnhancedSplitting" module to send EDRs to different output streams based on rules that you define. For example, you can split EDRs from roaming outcollects and incollects into different streams. See "Using Rules to Send EDRs to Different Output Streams".
- Use the "IRL_EventTypeSplitting" iRule to send EDRs to different output streams based on the service. See "Sending EDRs to Pipeline Output Streams".
- Use the "FCT_AccountRouter" to send EDRs to the correct pipeline in a multischema system. See "Using Pipeline Manager with Multiple Database Schemas".

You set up EDR preprocessing by configuring modules in a pipeline. For information on configuring preprocessing modules, see "Configuring EDR Preprocessing".

About Enrichment Modules

When you enrich EDR data, you add or change data for further processing.

- Use the "FCT_CliMapping" module to specify that multiple phone numbers owned by one customer are charged for on one bill. See "Mapping Multiple Phone Numbers to a Single Number".
- Use the "FCT_SocialNo" module to identify "social numbers" that are not displayed on an invoice. See "Setting Up Social Numbers".
- Use the "FCT_NOSP" module to identify a network operator and service provider when performing segment rating. See "Identifying the Network Operator/Service Provider ".
- Use the "FCT_NumberPortability" module to process events correctly when a customer changes network provider but keeps the phone number. See "Setting Up Number Portability".
- Use the "FCT_PrefixDesc" module to specify how to identify the destination of a call based on the number prefix. See "Creating Call Destination Descriptions".

You set up EDR enrichment by configuring modules in a pipeline. For more information about enrichment modules, see "Setting Up EDR Enrichment".

About Service Mapping Modules

Incoming EDRs from multiple switches often use different codes to represent the same service or supplementary service. To process EDRs, you need to normalize that data by mapping external service codes to internal service codes, service classes, and usage classes. You use the following modules:

- Use the "FCT_ServiceCodeMap" module to map external service codes to internal service codes. For example, CDRs might use different codes for different types of telephony services. You can map those codes to a single code; for example, TEL. The service is typically a bearer or primary service.
- Use the "FCT_UsageClassMap" module to map external codes for supplementary services, such as call forwarding, to internal usage classes. Usage classes are typically used for rating based on quality of service, sub-services, or service-level agreement values.
- Use the "IRL_UsageType" iScript to assign usage types to EDRs. A usage type is an internal code that represents an attribute of a customer account; for example, a friends and family discount.
- Use the "FCT_UoM_Map" module to convert the unit of measurement (UoM) of an incoming EDR to a UoM needed for rating a particular service. For example, an EDR might include the usage amount in seconds, but the service is configured to charge by minutes. The FCT_UoM_Map module converts seconds to minutes, using rounding rules.

You configure service mapping by configuring mapping modules in a pipeline.

About Zoning Modules

Zoning function modules prepare EDRs for rating by identifying geographic or logical zones that are used for rating.

You can identify zones as follows:

- Using logical source and destination; for example, area codes, the service used, or retail or wholesale usage.
- Using geographic distance.

Zoning is performed by the following function modules:

- Use the "FCT_PreRating" module to calculate zones and impact categories.
- Use the "FCT_APN_Map" module to process APN data for zoning.
- Use the "FCT_USC_Map" module to refine impact categories based on service attributes.
- Use the "FCT_Zone" module to compute zones when you use a pipeline only for zoning.

About Rating Modules

Rating modules rate EDRs. Each module performs a specific function; for example, the FCT_Dayrate module determines the time of day used for rating. Because each module adds or modifies data, the order of modules is important. For example, the FCT_RateAdjust module must process EDRs after the FCT_MainRating module. For more information, see "Function Module Dependencies".

- Use the "FCT_GlobalRating" module to apply a global charge to all EDRs.
- Use the "FCT_CustomerRating" module to provide customer data for other rating modules.
- Use the "FCT_RSC_Map" module to perform rating based on the quality of service when you set up service-level agreements (SLAs).
- Use the "FCT_RateAdjust" module to adjust charges after rating.
- Use the "FCT_BillingRecord" module to consolidate billing information for loading into the BRM database.

About Discounting Modules

Discounting modules run after rating modules. Pipeline discounting uses the following processes:

- The module checks for the conditions that allow a discount; for example, using 100 minutes per month.
- The module grants the discount; for example, reducing the charge by 10%.

You set up discounting by configuring the following discounting modules in a pipeline:

- Use the "FCT_DiscountAnalysis" module to analyze EDRs before discounting.
- Use the "FCT_Discount" module to calculate and apply discounts.

About Roaming Modules

Roaming modules rate roaming usage. You set up roaming rates by configuring the following modules in a pipeline. Use the "FCT_CarrierIcRating" module to supply the interconnect charge for the FCT_MainRating module.

About iScripts and iRules

Use iScripts and iRules to create custom pipeline functionality. For example, you can perform additional processing on data or add data to EDRs.

Use iScripts to perform the same operation on every EDR. Use iRules when you need to run an operation only under certain conditions.

How Pipeline Manager Uses BRM Data

Pipeline Manager needs to get data from the BRM database to rate each account. For example, Pipeline Manager gets the charge to use from the services and products/ charge offers owned by an account. Pipeline Manager also gets historical data; for example, if a customer changes a phone number, Pipeline Manager needs the old number to rate calls made using it.

Pipeline Manager also needs to get data that is not required for rating but is required by the BRM event objects that Rated Event (RE) Loader loads into the BRM database. For example, every event requires an item, so Pipeline Manager needs to get the correct item for the event. When the rated event is loaded, the correct item is already recorded in the event.

How Pipeline Manager Identifies Accounts

When Pipeline Manager rates usage events, there is no information in the original CDR that specifies which BRM account was responsible for the event. Pipeline Manager needs to know the account to rate the event and to apply discounts.

To find the account:

1. In the EDR, Pipeline Manager finds the phone number identified as the calling number.
2. In the data retrieved from BRM, this number is stored in the alias list in a service object. Once the service object is found, it has a pointer to the account object, from which Pipeline Manager identifies the account.

How Pipeline Manager Chooses a Charge

Pipeline Manager needs a charge for determining usage charges. There are three ways that Pipeline Manager can find out which charge to use:

- **Product/Charge offer priority**. By default, Pipeline Manager uses the charge associated with the highest-priority product/charge offer. That is, it searches through all purchased products/charge offers, from highest priority to lowest priority, until it finds one that matches the event's rating criteria, such as the zone model and time model type. It then selects the charge associated with that charge offer. You assign priorities to products/charge offers.
- **Lowest charge**. When configured for least cost rating, Pipeline Manager finds the product/ charge offer that generates the lowest overall charge to the customer and then uses the charge associated with the product/charge offer. You configure a pipeline for least cost rating by using the IRL_LeastCostPerEDR and ISC_LeastCost modules. See "About Least Cost Rating".
- **ERA**. If a service or an account is associated with an extended rating attribute (ERA), Pipeline Manager uses the charge you configure and prioritize for the ERA.

> **Note:**
>
> If a subscription service and member service both own a service-level ERA, the member service's ERA has priority and is used for selecting the charge.

How Pipeline Manager Assigns Delayed Events to Items

When Pipeline Manager outputs events to RE Loader, the events must include all mandatory event data. Most of that data comes from the incoming CDR; for example, the call origin and destination. Some of the data must come from BRM, including which bill item stores the balance impact of the event.

In BRM, every event object is associated with a bill item. The incoming CDR does not have any information about bill items, so Pipeline Manager gets that information from BRM.

1. In the EDR, Pipeline Manager finds the phone number identified as the calling number.
2. In the data retrieved from BRM, this number is stored in the alias list in a service object. Once the service object is found, Pipeline Manager uses the information in the item POID list to determine which bill item the event applies to.

BRM creates service usage items for the next accounting cycle on four occasions:

- When the service is purchased by an account.
- When billing is run.
- When the Bill Now feature is used in Billing Care.
- When a usage event occurs.

However, BRM does not create a usage item when Pipeline Manager processes an event. If no usage item exists, Pipeline Manager rejects the event. Therefore, BRM pre-creates usage items for Pipeline Manager to use.

BRM pre-creates items in the following cases:

- When an account is created.
- When you run billing.
- When a CSR uses Bill Now.
- At the end of the accounting cycle when you use delayed billing.

> **Note:**
>
> To maintain correct discount balances, and to minimize rejected EDRs, always use delayed billing when you use pipeline rating.

To choose the correct bill item, Pipeline Manager can do one of the following:

- Assign the event to the current open bill item.
- Assign the event to the next open bill item.
- Reject the event because it belongs to an item that has already been included in a bill.

To decide which item to apply the bill to, Pipeline Manager takes into account the following dates:

- The date when the call occurred (the EDR date).
- The current system date.
- The date when the current accounting cycle ends. This is called the *next accounting cycle* date.

- The number of days after the current accounting cycle ends when delayed billing runs. This number is called the *delayed billing offset*.

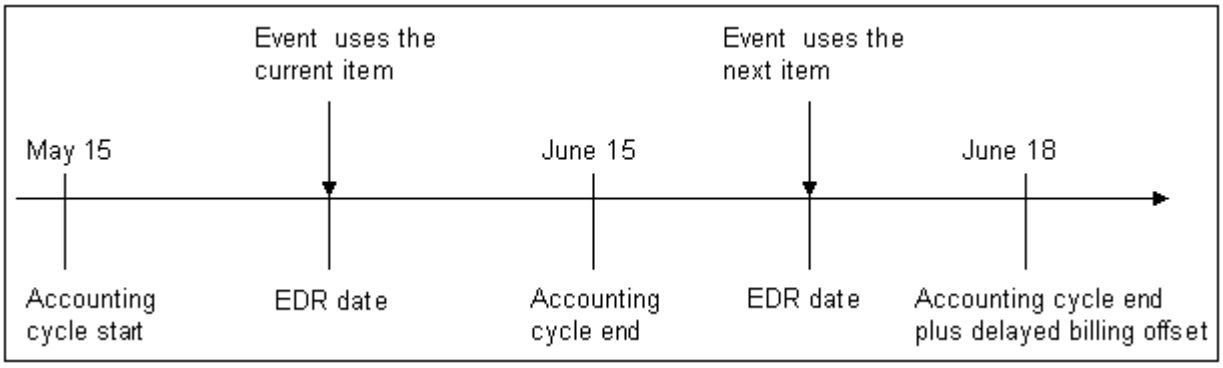
> **Note:**
>
> You can set up an accounting cycle delay period to manage which items delayed events are assigned to. This is useful when your billing cycle spans multiple accounting cycles. See "About Accounting Cycle Delay Periods".

To assign the event to an item:

- If the EDR date falls before the next accounting date, the event is assigned to the current item.
- If the EDR date falls after the next accounting date, the event is assigned to the next item. This can happen because the event might occur after the close of the accounting cycle but before the delayed billing offset date.

Figure 17-4 shows how events are assigned:

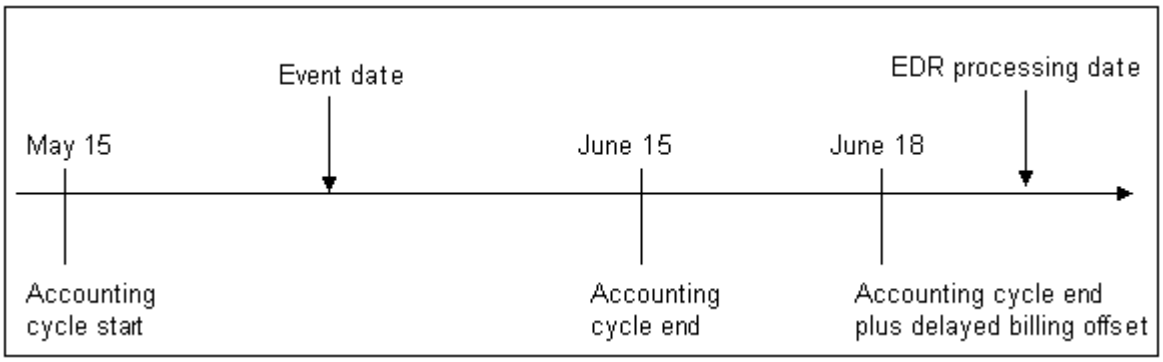
Figure 17-4 Event Assignment



> **Note:**
>
> The customer billing date is not relevant when choosing which item to use for the event. There might be multiple accounting cycles in one billing cycle. New items are created for each accounting cycle.

If Pipeline Manager needs to assign an event to the current item, but billing for that item has already occurred, Pipeline Manager includes the event and assigns it to the current item. For example, if the event is rated on May 20 and loaded after the account cycle ends, it is still included in the current item as shown in Figure 17-5.

Figure 17-5 Current Event Assignment After Billing



About Accounting Cycle Delay Periods

In the batch pipeline, events that occur in one cycle can sometimes be rated and loaded into the BRM database after that cycle is completed. To assign delayed events to items of the billing cycle in which they occurred, you configure delayed billing. To assign delayed events to items of the accounting cycle in which they occurred, you configure an accounting cycle delay period.

When a billing cycle spans multiple accounting cycles, the items for those accounting cycles are not closed until billing is run. If you run a general ledger (G/L) report at the end of an accounting cycle for which billing has not yet been run, the status of the revenue in the G/L can change if additional events are rated and loaded for that cycle before the accounts are billed.

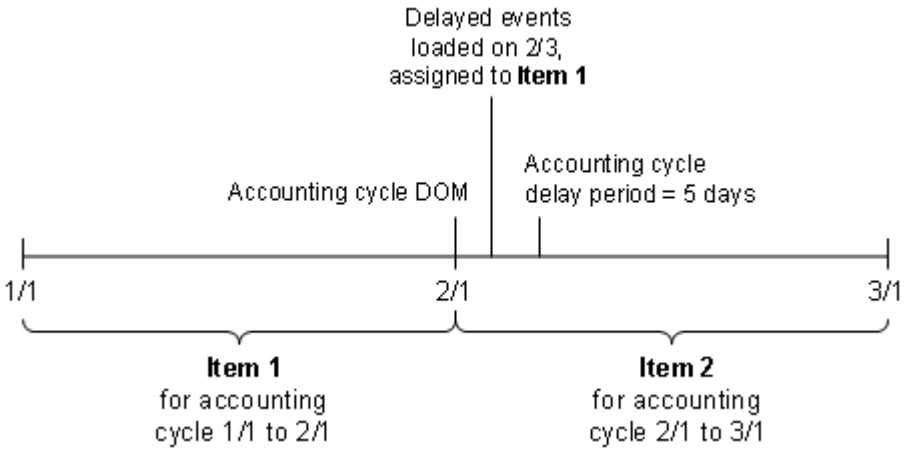
If you require that G/L data not change after the G/L report is run, you can configure an accounting cycle delay period after which events are no longer assigned to items of that accounting cycle, even if those items are not closed. You then run G/L reports after the accounting cycle delay period has ended. This ensures that the revenue reported in the G/L is accurately represented and that the state of the revenue (earned, unearned, billed, and unbilled) does not change after the G/L report is run.

> **Note:**
>
> To use an accounting cycle delay period, you must also configure delayed billing.

When you configure an accounting cycle delay period, BRM assigns delayed events to items based on when the accounting cycle delay period ends. BRM assigns events to items when RE Loader loads the events into the database:

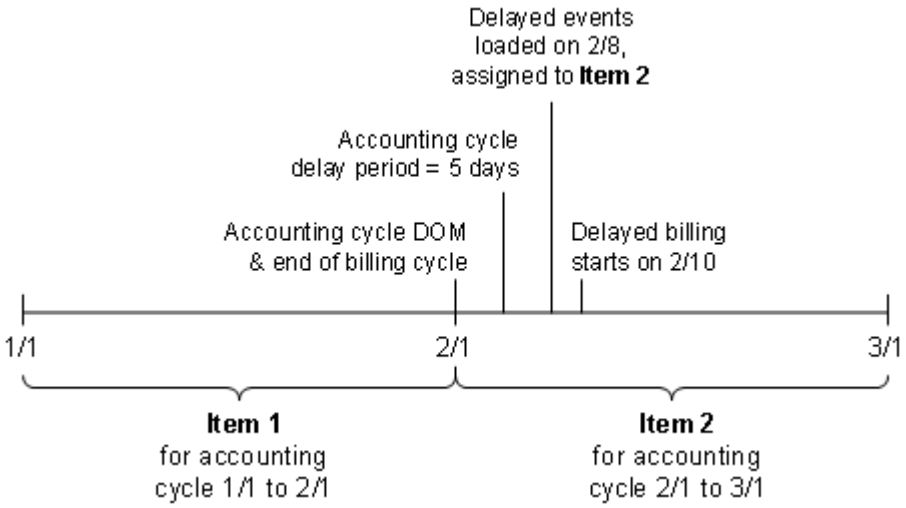
- When delayed events (events that occurred in the previous cycle) are loaded after the accounting day of month (DOM), but before the end of the accounting cycle delay period, those events are posted to the item for which the DOM has just passed as shown in Figure 17-6:

Figure 17-6 Delayed Events Arriving During Cycle Delay Period



- When the billing cycle has ended and delayed events are loaded after the end of the accounting cycle delay period, but before delayed billing is run, those events are posted to the item for the next (current) accounting cycle, even though the previous cycle has not been billed and its items are still pending. This is shown in Figure 17-7:

Figure 17-7 Delayed Events Arriving After Cycle Delay Period



- After the account is billed, items for the billed cycle are closed so delayed events are posted to the item for the following (the current) cycle.

> **Note:**
>
> If the accounting cycle delay period is longer than the delayed billing period, the accounting cycle delay period is ignored after billing is run. After billing is run, if any remaining events that occurred in the previous cycle are rated and loaded in the current cycle, they are assigned to the current cycle's item.

You specify the accounting cycle delay period by modifying a business parameter configuration (**/config/business_params** object). RE Loader checks the accounting cycle delay period in the **/config/business_params** object before loading events. See "Configuring an Accounting Cycle Delay Period".

Configuring an Accounting Cycle Delay Period

By default, an accounting cycle delay period is disabled. You can enable this feature by modifying a field in the **billing** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility.

To configure an accounting cycle delay period:

1. Use the following command to create an editable XML file from the **billing** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. The file contains the current billing configuration values in the **/config/business_params** object in the BRM database. If you don't want this file in your working directory, specify the path as part of the file name.

2. Open **bus_params_billing.xml.out** file and search for the following line:

   ```
   <AcctCycleDelayPeriod>-1</AcctCycleDelayPeriod>
   ```

3. Change **-1** to the number of days in the accounting cycle delay period. The number of days must be a positive value. (A value of **-1** indicates that there is no accounting cycle delay period.)

   For example, if the accounting cycle delay period is 3 days and the accounting cycle ends at midnight on March 31, the delay period ends at midnight on April 3.

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

   > **Note:**
   >
   > Do not set the accounting cycle delay period to be longer than the delayed billing period.

4. Save and close the **bus_params_billing.xml.out** file and rename the file to **bus_params_billing.xml**.

5. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

You should execute this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. *BRM_home* is the directory where you installed BRM components.

6. Read the object by using the **testnap** utility or Object Browser to verify that all fields are correct.
7. Stop and restart the Connection Manager (CM).

About G/L IDs

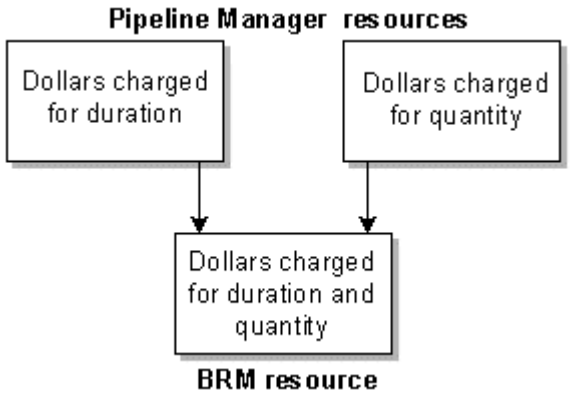
Pipeline Manager uses G/L IDs defined in the Pipeline Manager database. However, those G/L IDs must match the G/L IDs in the BRM database. You should define G/L IDs in the BRM database first and then in Pipeline Manager.

About Mapping Balance Elements between the Pipeline Manager Database and the BRM Database

Balance Element IDs must match between the Pipeline Manager database and the BRM database. In addition, you can configure how Pipeline Manager balance elements map to BRM balance elements. For example, Pipeline Manager can separate balance element amounts in a variety of ways, but you might want to combine balance element amounts when defining BRM balance impacts.

Figure 17-8 shows how the charges for a GPRS event can be mapped between the Pipeline Manager database and the BRM database:

Figure 17-8 GPRS Charge Mapping Between BRM and the Pipeline Manager



How Pipeline Manager Gets Historical Data

Because there is a gap of time between when a call occurs and when it is rated, information about the customer can change during that time. For example, a customer might change the phone number before a call is rated. Pipeline Manager needs to look up account data based on the old number.

To retrieve historical information, Pipeline Manager gets data from audited objects. By default, auditing in BRM is turned off for most objects. After you install the Account Synchronization DM, you must run the **object_auditing.pl** script to turn on auditing for the objects and fields that Pipeline Manager needs data about.

In addition, Pipeline Manager pricing configuration data includes validity dates that can be used to apply the correct rating to delayed events.

About Loading Pipeline-Rated Event Data

Pipeline Manager sends the results of rating to output files. BRM loads the data in these files into the BRM database.

You configure BRM Batch Controller to start a batch handler when rated event files are ready for loading.

You configure a batch handler to load the rated-event data into the BRM database. The batch handler runs the utilities that load the data.

BRM provides the following sample batch handlers that you can use to load pipeline-rated events:

- **SampleRelHandler**: This batch handler starts the Rated Event (RE) Loader utility (pin_rel). The utility loads data for events that are rated in batches.
- **OODHandler**: This batch handler starts the rerate-request loader utility (pin_load_rerate_jobs). The utility creates rerate jobs for events that were rated out of order.
- **SampleHandler**: This batch handler starts the Universal Event (UE) Loader utility (uel). The utility loads batches of events into the BRM database for rating by the rating opcodes. UE Loader is also used by Pipeline Manager to load various types of data files that it generates during rating; for example, validity period data for products/charge offers, discounts/discount offers, and balance elements that start on first usage, roaming settlement data, and revenue assurance data.
- **ConfigurableValidityHandler**: This batch handler starts the **pin_rel** and **pin_load_rerate_jobs** utilities, and the instance of **uel** that loads first-usage validity data, thereby eliminating the need to configure separate instances of the handlers for these utilities. For more information, see "About Using a Single Batch Handler to Run Multiple Loading Utilities".

About Using a Single Batch Handler to Run Multiple Loading Utilities

If you use Pipeline Manager to rate products/charge offers, discounts/discount offers, or granted balance elements that start on first usage, you configure the ConfigurableValidityHandler sample batch handler to run the utilities for RE Loader (pin_rel), UE Loader (uel), and out-of-order rerating (pin_load_rerate_jobs).

> **Note:**
>
> If you do not wish to use a single batch handler to run these utilities, you can configure the individual sample batch handlers provided for RE Loader, UE Loader, and out-of-order rerating.

ConfigurableValidityHandler runs **pin_rel**, **uel**, and **pin_load_rerate_jobs** instances sequentially, waiting for one process to complete before starting the next. In a single sequence, ConfigurableValidityHandler processes files that were produced for the same CDR file or for the same pipeline transaction (if the pipeline is configured to produce an output file per transaction instead of per CDR file).

When pipeline rating outputs a file of rated events, the ConfigurableValidityHandler batch handler performs the following tasks:

1. Runs **pin_rel** to load the results of pipeline rating into the BRM database.

   For more information about RE Loader, see "Understanding Rated Event Loader".

2. Checks for first-usage validity files for products/charge offers and discounts/discount offers and, if one exists for the same CDR or transaction, runs **uel** to load the product/charge offer and discount/discount offer validity data.

   For more information about using UE Loader to load validity data, see "About Updating Validity Period Information in the BRM Database".

3. Checks for first-usage files for balance elements and, if one exists for the same CDR or transaction, runs **uel** again to load the balance element validity data.

4. Checks for rerate-request files for events rated out of order and, if one or more exist for the same CDR or transaction, runs **pin_load_rerate_jobs** for each file.

ConfigurableValidityHandler handles errors in the following ways:

- If **pin_rel** does not successfully load the pipeline batch-rated events, the failure is logged in the handler log file (**configurable_validity_handler.log**) and **uel** and **pin_load_rerate_jobs** are not run.
- If **uel** or **pin_load_rerate_jobs** fails, the entire process is recorded as failed in the handler log file.

If all processes are successful, the **configurable_validity_handler.log** file is deleted. If one or more processes are not successful, the **configurable_validity_handler.log** file is not deleted.

Along with processing information and status, the log files include the name of the input file that was loaded. If **pin_rel** loads a rated-event file and there were no associated first-usage validity files or rerate-request files, this is noted in the log file.

About Pipeline Rating and BRM Billing

When Pipeline Manager receives an EDR for the next billing cycle for an account that hasn't yet been billed, the EDR is suspended (not rated). Only when the account's billing process is complete can the new EDRs be rated.

The number of accounts being billed affects the time it takes to complete the billing process. The longer the processing time, the greater the chance EDRs might need suspending or rerating. If you process many EDRs and need accounts to be billed quickly so that their new usage can be rated, you can set up Pipeline Manager to trigger billing. When Pipeline Manager triggers billing for an account, it is billed in a separate billing process.

Function Module Dependencies

Table 17-1 provides guidelines for how to configure the order of function modules in a pipeline. The modules are listed in a typical order, but your configuration might vary. Some modules require that other modules be run first, whereas some modules can be located anywhere in the pipeline.

For more information, see the reference documentation for the module.

Table 17-1 Functional Modules and Processing Dependencies

Function module	Processing dependencies
FCT_PreSuspense	Must be the first preprocessing module in a pipeline.
ISC_SetAndValidateBatchInfo	This is the first iScript that must be used so that the batch ID gets inserted before any further processing of the mediation batches.
FCT_DuplicateCheck	Should run early in a pipeline to discard duplicate EDRs.
FCT_CallAssembling	Should run early in a pipeline to assemble EDRs. Must run before FCT_Discard.
FCT_ServiceCodeMap	Some modules require an internal service code, so this module should run near the front of a pipeline.
ISC_CiberInputValidation	Because erroneous CIBER records can be discarded, this module must run before the FCT_Discard module.
FCT_Discard	Because you can discard or split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap module. Should be early in the function pool, but must be run after FCT_CallAssembling.
iRuleValidation	Run this iRule before ISC_TapSplitting.
ISC_TapSplitting	Must run after the following modules: • FCT_DuplicateCheck • FCT_Discard
FCT_AccountRouter	For general use, this module must run after the FCT_ServiceCodeMap module and before the rating modules. For use with standard recycling or Suspense Manager using multiple database schemas, this module must run before OUT_GenericStream in a pre-recycling pipeline. This module sends output to a separate pipeline for each BRM database schema.
FCT_EnhancedSplitting	Because you can split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap and FCT_UsageClassMap modules.
FCT_CliMapping	Must run before the rating modules.
FCT_UoM_Map	Must run after the FCT_ServiceCodeMap module and before the rating modules.
FCT_UsageClassMap	Must run before the zoning and rating modules.
FCT_NumberPortability	Must run before the zoning and rating modules.
ISC_MapNetworkOperatorInfo	Must run after the FCT_NumberPortability module and the ISC_PopulateOpcodeAndUtilBlock_Diameter iScript.
FCT_NOSP	Must run before segment rating is performed.
FCT_Account	Must run before the zoning and rating modules.
ISC_ProfileAnalyzer	Must run after FCT_Account and before any rating modules.
ISC_ProfileLabel	Must run after FCT_Account and before any rating modules.
IRL_UsageType	Must run after FCT_Account and before FCT_USC_Map.
FCT_CiberOcc	Must run after the FCT_DuplicateCheck module and before the FCT_CarrierIcRating module.

Table 17-1 (Cont.) Functional Modules and Processing Dependencies

Function module	Processing dependencies
FCT_ItemAssign	Must run after the FCT_Account, rating, and discounting modules and before the FCT_BillingRecord module.
FCT_CustomerRating	Must run after FCT_Account.
FCT_Filter_Set	Must run after FCT_Account.
IRL_PromotionalSavingPerEDR	Must run before IRL_LeastCost and FCT_CustomerRating.
IRL_LeastCostPerEDR	Must run before FCT_CustomerRating and *after* "FCT_Filter_Set".
FCT_CustomerRating	Must run after FCT_Account.
ISC_LeastCost	Must run after FCT_CustomerRating.
FCT_CiberOcc	Must run after the FCT_DuplicateCheck module and before the FCT_CarrierIcRating module.
FCT_CarrierIcRating	Must run after FCT_Account.
FCT_DroppedCall	Must run after FCT_Account.
FCT_APN_Map	Can run before or after the zoning modules (FCT_Zone and FCT_PreRating).
FCT_PreRating	Must run before the FCT_MainRating module.
FCT_SegZoneNoCust	Must run after the prerating module
FCT_MainZoning	Must run after the FCT_SegZoneNoCust module.
FCT_Zone	Must run after FCT_Account.
FCT_USC_Map	Must run after the following: • FCT_UsageClassMap • ISC_UsageType • FCT_PreRating
FCT_TriggerBill	Must run before the FCT_MainRating module.
FCT_MainRating	Must run after at least one of the following modules: • FCT_GlobalRating • FCT_CustomerRating • FCT_CarrierIcRating
FCT_RSC_Map	Must run after the FCT_MainRating module to adjust the rate.
FCT_Dayrate	Must run after the FCT_MainRating module to adjust the rate.
FCT_RateAdjust	Must run after the FCT_MainRating module to adjust the rate.
FCT_Rounding	Must run after the FCT_RateAdjust module if you want rating results to be rounded and after FCT_Discount module if you want discount results to be rounded. FCT_Rounding must come after each module for which rounding should occur. For batch rating, it must come before the FCT_ApplyBalance module.
FCT_ExchangeRate	Must run after the FCT_MainRating module.
ISC_FirstProductRealtime	Must run this iScript in the real-time rerating pipeline before the FCT_DiscountAnalysis module.

Table 17-1 (Cont.) Functional Modules and Processing Dependencies

Function module	Processing dependencies
FCT_DiscountAnalysis	For pipeline rating, must run after the FCT_Account module and before the FCT_Discount module. For real-time rating, must run before the FCT_Discount module.
FCT_Discount	Must run after the FCT_MainRating module. For batch rating, this module should be in the same function pool as the FCT_ApplyBalance module and run before that module.
FCT_FirstUsageNotify	Must run this module before the FCT_ApplyBalance and FCT_Reject modules.
FCT_ApplyBalance	Must run after the FCT_Rounding module. This module should be in the same function pool as the FCT_Discount module and run after that module.
ISC_TaxCalc	Must Run this module before the FCT_BillingRecord module, but after the FCT_MainRating module.
FCT_BillingRecord	Must run after the FCT_MainRating and FCT_Discount modules.
FCT_EventOrder	Must run *after* the FCT_MainRating and FCT_Discount modules and *before* the FCT_Reject module.
ISC_CiberOutputMapping	Must run after the FCT_MainRating module and ISC_PostRating iScript.
IRL_EventTypeSplitting	Must run after FCT_ServiceCodeMap and before the FCT_Reject module. This is typically the last module before the FCT_Reject module.
FCT_Reject	Runs after the rating and discount modules.
ISC_SetOutputStream	Must run after FCT_Reject.
ISC_PostRating	Must run: • After rating modules FCT_CustomerRating, FCT_PreRating, and FCT_MainRating or • After the FCT_ExchangeRate module
ISC_SetEDRStatus	Must run before FCT_AggreGate.
ISC_SetRevenueFigures	Must run after rating and discounting and before FCT_AggreGate.
ISC_SetRevenueStream	Must run before FCT_AggreGate and after post rating (after the EDRs are rated).
ISC_SetEDRStatus	Must be run before the FCT_AggreGate scenario that collects audit data grouped on the EDRStatus field.
FCT_AggreGate	Runs after rating modules. This module usually runs in its own pipeline.
FCT_CancelTimer	Depends on the FCT_Timer in the Dispatcher pipeline for the TimerID and the timeout flag values.
FCT_Recycle	For pipeline-only recycling, must be the last module in the pipeline.

Table 17-1 (Cont.) Functional Modules and Processing Dependencies

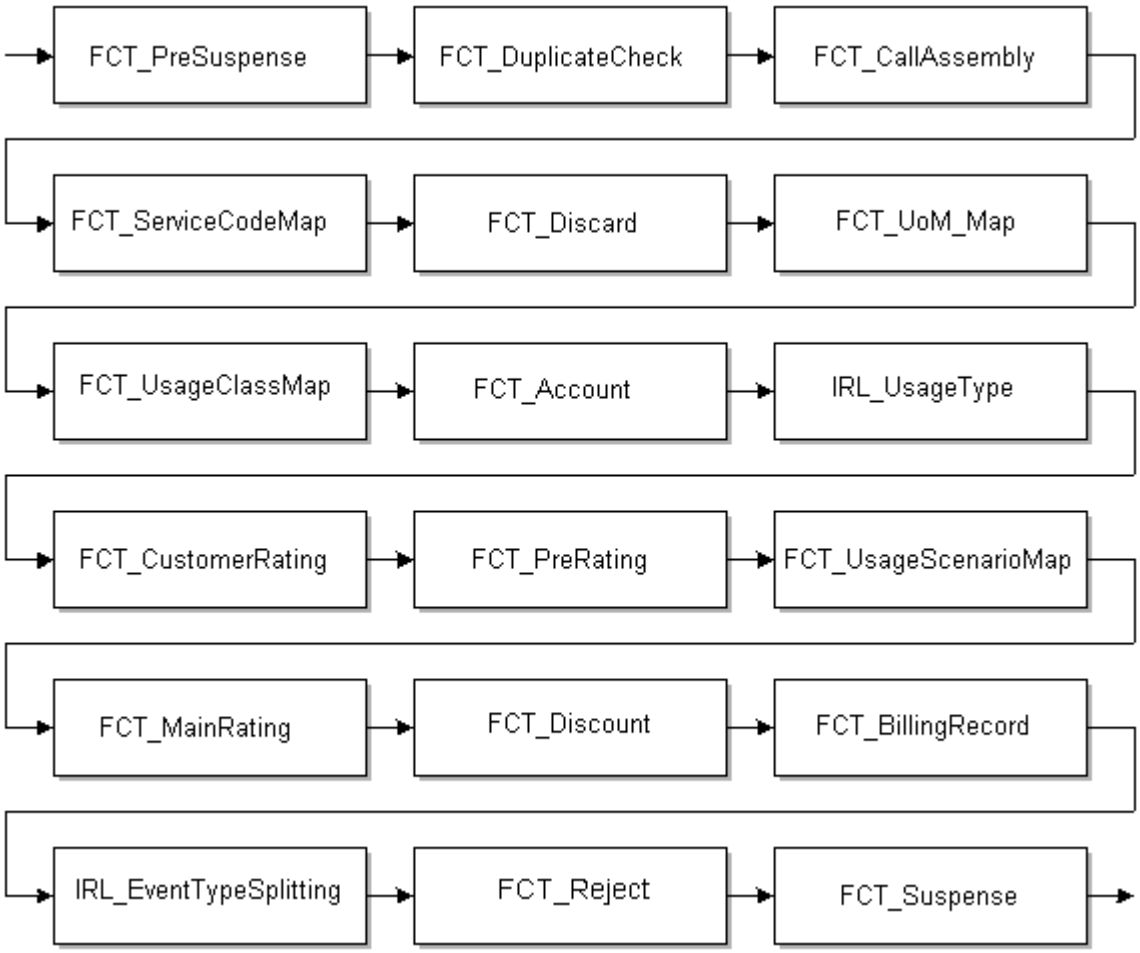
Function module	Processing dependencies
FCT_Suspense	For standard recycling and Suspense Manager, must be the last module in a pipeline.

The following modules do not have any placement dependencies and can be run anywhere in a pipeline or in a separate pipeline, depending on the data that is being processed:

- FCT_IRules
- FCT_IScript
- ISC_AddCBD
- IRL_PipelineSplitting
- FCT_Timer
- FCT_PreRecycle
- FCT_Opcode
- ISC_ApplyTax
- ISC_BACKOUTTypeSplitting
- ISC_CiberRejectReason
- ISC_ConsolidatedCP
- ISC_DupRAPRecords
- ISC_EDRToTAPOUTMap
- ISC_Migration
- ISC_MiscOutcollect
- ISC_ObjectCacheTypeOutputSplitter
- ISC_OverrideSuspenseParams
- ISC_RemoveASSCBD
- ISC_RollbackSettlement
- ISC_UsageClassSetting
- FCT_SocialNo
- FCT_PrefixDesc

Figure 17-9 shows the order of the most common function modules:

Figure 17-9 Order of Common Function Modules



Data Module Dependencies

This section provides guidelines for how to configure the order of data modules in the registry file. Some modules require that other modules be run first, whereas some modules can be located anywhere in the pipeline.

- "DAT_PortalConfig". Due to the dependency of other data modules on DAT_PortalConfig, the DAT_PortalConfig registry entries must appear before all other data module entries in the registry file.

18

Policy-Driven Charging

This chapter describes how to manage policy-driven charging of services by using Oracle Communications Billing and Revenue Management (BRM) real-time and offline rating.

Topics in this document:

About Policy-Driven Charging in BRM

Policy-driven charging enables you to track a subscriber's service usage and, based on that usage, change the customer's quality of service (QoS) in real-time rating and in offline rating for offline events such as billing and adjustments.

For example, a subscriber purchases a plan for a certain QoS to download video content. The subscriber makes his choice from one of many plans that you have configured with gradations in the QoS based on usage amounts in megabytes, such as 100-150, 150-200, and 200-250 megabytes. When the subscriber starts downloading video content from the network, you can track the quantity of megabytes that the subscriber downloads during that session. When you find that the downloaded quantity crosses the upper threshold set for the selected QoS (for example 150 megabytes), you can use BRM's policy-driven charging to make a seamless change in the policy set for the subscriber's (video downloading) session on the network and allow a change to the QoS from the current to the next level.

About Setting Up Policy-Driven Charging

To set up policy-driven service charging:

- Create offer profiles to define ranges in the QoS for the services you offer, and include the offer profiles in your price lists.

  For example, set up high, medium, and low QoS for downloading data from the network, using a noncurrency resource called *Megabytes Used*. Create an offer profile with a unique name (for example, *Platinum*). In the offer profile, specify the resource ID **100009** for the noncurrency resource and set the ranges to 100-150, 150-200, and 200-250 megabytes.

- Link the offer profile to the product or discount with which it is associated so that BRM can track the usage correctly.

  For example, connect the offer profile called *Platinum* to a product or discount by using *Platinum* as the provisioning tag for the product or discount. Use the resource ID **100009** for the noncurrency resource *Megabytes Used* configured in the offer profile as the noncurrency resource in the rate plan for the product or discount.

- Set up provisioning rules that state how the network is to provide and manage the services you offer.

  For example, when a subscriber's usage of a video downloading service reaches or crosses a threshold set in the offer profile (for example,150 megabytes of data download), a different rule can be enforced for the network session, based on the subscriber's profile, his balances, his usage of the resource, and the traffic on the network.

  You configure these provisioning rules in the application you use to provide network policy control. Policy controllers (such as Oracle Communications Policy Controller) manage the QoS, optimize high bandwidth traffic, and enforce usage quota levels for subscribers using the network.

  The policy controller installs the provisioning policy on the network element responsible for managing the enforcement of network policies on traffic passing through the network.

> **Note:**
>
> Setting up provisioning rules to use in policy-driven charging for network capacity and services is beyond the scope of this document. For more information, see the appropriate references.

In addition to configuring offer profiles for provisioning rules in the policy controller, you must configure the following:

- Subscriber preferences for communications, such as whether the subscriber wants to be notified, in what language, and so on.
- Other factors (such as rates) that you use to give better services to more valuable subscribers when network capacity is limited during peak time or in an area.
- A network connectivity application to start, monitor, and end the online charging services for sessions in the network.

  Network connectivity applications (such as Oracle Communications Online Mediation Controller) communicate with BRM to manage charging sessions.

> **Note:**
>
> Setting up the network connectivity application for policy-driven charging is beyond the scope of this document. For more information, see the appropriate references.

- Event notification to trigger policy-driven charging operations for real-time charging.

How Policy-Driven Charging Works

BRM uses the data you configure in offer profiles to provide you with offer profile threshold breach notifications for both in-session and out-of-session rating events.

Suppose you configure an offer profile called *Platinum* with specific usage ranges for downloading video content. The noncurrency resource such as *Megabytes (MB) Used* and its usage ranges such as *100-150* megabytes, at *150-200* megabytes, over *200-250* megabytes are defined within a policy label (called *Fair Usage*).

When your subscriber initiates the downloading of a video,

- The policy controller communicates with BRM to set up a provisioning policy for the session.

  A provisioning rule is established on the network for the session for an initial quantity, for example, 35 megabytes. The range for the current QoS for the subscriber is 100-150 megabytes.

- The download begins with the network connectivity application managing the online-charging session.

Throughout that session, whenever the subscriber requests more quota, BRM calculates whether it can allocate the requested amount or if it must readjust the quota.

For example, BRM receives a request with the information that the subscriber has used 35 megabytes and needs 75 megabytes more. BRM does the following:

- Calculates how much the subscriber has used.

  BRM uses the balance information and consumed reserved amount (across parallel sessions, iPhone, video, and computer) for that subscriber. For example, BRM calculates that the subscriber has used a total of 150 megabytes. With 100-150 megabytes as the range for the current QoS for the subscriber, his usage is at the threshold for the policy.

- Reduces the allocation, if necessary.

  In the example case, the subscriber has used a total of 150 megabytes. The upper thresholds of the offer profiles are set at 150, 200, and 250 megabytes. The next threshold is 200 megabytes. The available quota for that QoS is 50 megabytes only (200-150). The subscriber has requested 75 megabytes, an amount which is greater than the available quota of 50 megabytes.

  In this scenario, BRM readjusts the quota allocation to 50 megabytes.

When the used amount for the resource crosses a set threshold (150, 200, 250, and so on), the provisioning rule (which was tied to the previous policy level) can no longer be applied for the session.

BRM and the policy controller ensure that a more appropriate rule is established for the session on the network in the following way:

- BRM sends an offer profile threshold breach notification to the policy controller with information on the threshold breach (150 in our example case) and information on the subscriber's preferences for receiving communications.
- The policy controller does the following:
  - Communicates with BRM to retrieve the latest information on the subscriber and his usage of the service.
  - Arrives at the new rule which enables the increase in bandwidth.
  - Establishes that provisioning rule for the session.

About Notifications Related to Offer Profiles Sent by BRM

When you set up policy-driven charging for your services, BRM sets up notification events for the following:

- When you create, modify, or delete the following:
  - Offer profiles
  - Products associated with offer profiles
  - Discounts associated with offer profiles
- Offer profile threshold breaches

  Offer profile threshold breaches can occur with your subscriber's usage of a resource in a prepaid session. Or, they can occur when BRM processes offline events, such as billing and adjustments.

BRM places these offer profile threshold breach notifications in a central Oracle Advanced Queuing (AQ) database queue for use by customer relationship management (CRM) applications. You can use these applications to retrieve data directly from the Oracle AQ database queue and use the information in these offer profile threshold breach notifications. See *BRM Developer's Guide* for more information on the Oracle AQ database queue.

BRM provides the following information in offer profile threshold breach notifications associated with policy-driven charging:

- Service identifier
- List of aliases that also identify this user/service (ALIAS_LIST)
- Name of the resource
- Resource ID
- Status label from the offer profile

See "Sample Threshold Breach Notification".

Configuring Your BRM Environment to Support Policy-Driven Charging of Services

Configure your BRM environment to support policy-driven charging of services by completing the following if you are using Pricing Center and Pipeline Manager:

1. Set up offer profiles in your price lists. See "About Setting Up and Managing Offer Profiles".
2. Configure BRM to track a subscriber's usage of the resources you offer. Set up BRM to create notifications tailored to the subscriber's preferences on how they want to be notified

when their usage exceeds the threshold set in their offer profiles. See "About Configuring Resources for Tracking by BRM ".

3. Merge the event notification entries specific to policy-driven charging with the BRM event notification list. See "Configuring Event Notification for Policy-Driven Charging".
4. Enable your enterprise applications to integrate with policy-driven charging in BRM. See "Enabling Enterprise Applications to Handle Policy-Driven Charging Events".
5. Update the BRM database with reservation preferences configuration for policy-driven charging. See "Updating Reservation Preferences Configuration for Policy-Driven Charging".
6. Ensure that the Connection Manager can forward the notifications associated with policy-driven charging. See "Updating the Connection Manager for Policy-Driven Charging".

About Setting Up and Managing Offer Profiles

You can create offer profiles when you create a price list or add your offer profile data to an existing price list. See "Setting Up Offer Profiles" for information on setting up the data for offer profiles.

About Using New Resource IDs in Your Offer Profiles

When you start (or restart) your BRM system, BRM caches the set of unique resource IDs that you have configured in your offer profiles. BRM processes a subscriber's usage only if the resource ID is present in its cache.

> **Note:**
>
> When you introduce a resource ID that has not been used in BRM, you must update the entries in this cache. To do so, stop and start the Connection Manager (CM).

Configuring Offer Profile Data

BRM provides sample offer profile data in the default **pin_offer_profile_policy_labels.xml** file.

This sample file uses the format detailed in the *BRM_Home*/**PricingCenter/price_list.xsd** file. *BRM_Home* is the directory in which you installed the BRM software. Maintain your offer profiles in the format used in the sample file. For more information on using the XML interface, see "Using the XML Pricing Interface to Create a Price List ".

> **Note:**
>
> - Offer profile names must be unique.
> - If you introduce a new resource ID in an offer profile you add to or modify in the database, you must restart the CM after your load the offer profiles.

To configure your offer profile data:

1. Go to the *BRM_Home*/**setup/scripts** directory.
2. Open the **pin_offer_profile_policy_labels.xml** file in an XML editor or text editor.

> **Tip:**
>
> Save a copy of the default file before you make any changes to it.

3. Edit this file using the parameters as shown in the file. Table 18-1 describes the elements.

Table 18-1 Elements Used to Store Offer Profiles

Element	Description
label	Name of the policy label
offer_profile	Array to hold offer profiles
offer_profile_code	Key identifier associated with an offer profile
offer_profile_name	Name of an offer profile
policy_label	Array to hold policy labels
price_list	Price List object within which the offer profile array resides
resource_id	Id used for the resource associated with a policy label
resource_name	Name of the resource associated with a policy label
status_label	Status label name indicating the level of the tier
tier	The tier as an array with each entry made up of a status label, and the start and end of the range for that tier level.
tier_end_range	Start of the range for a tier level
tier_start_range	End of the range for a tier level
unit	Unit of the resource associated with a policy label. The values are: • None. (**0**) • Second (**2**) • Minute (**1**) • Hour (**3**) • Day (**4**) • Byte (**11**) • Kilobyte (**12**) • Megabyte (**13**) • Gigabyte (**14**) The default value is **0**.

4. Save the **pin_offer_profile_policy_labels.xml** file.

You are now ready to store the offer profiles in the BRM database.

Managing Offer Profile Data

BRM stores offer profiles in **/offer_profile** objects. Use the **loadpricelist** utility to load, retrieve, modify, and delete **/offer_profile** objects.

> **Note:**
>
> Before you use this utility, verify that:
>
> - The XML offer profile file is complete and follows the guidelines.
> - The **Infranet.properties** file in your CLASSPATH has the correct login information for the BRM database to which you want to connect.
>
> For more information, see "Prerequisites for the XML Pricing Interface".

Storing Offer Profile Data

> **Note:**
>
> If you introduce a new resource ID in an offer profile you add to the database, you must restart the CM after you load the offer profiles.
>
> To do so, stop and restart the CM.

Use the following command to load offer profiles into your BRM database:

```
loadpricelist -f -v -c pin_offer_profile_policy_labels.xml
```

where:

- **`-f`** runs the command without prompting for a confirmation.
- **`-v`** displays information about successful or failed processing as the utility runs
- **`-c`** reads the offer price information from **pin_offer_profile_policy_labels.xml** and commits it to the BRM database.

The offer profile information is now stored in **/offer_profile** objects in the database.

Loading Modified Offer Profiles in the Interactive Mode

You can commit the modified offer profile data by using the **loadpricelist** utility when you are working in the interactive mode.

To load offer profile data only:

```
write PriceListFile offer_profile PriceObject
```

where:

- *PriceObject* is the specified pricing object in the database.
- *PriceListFile* is the entire price list in the database.

For more information on the utility, see "loadpricelist".

Retrieving Offer Profiles

To retrieve offer profile data, use the **loadpricelist** utility in the following ways:

- Non-Interactive mode:

  ```
  loadpricelist -r PriceListFile
  ```

- Interactive mode:

  ```
  read PriceListFile
  ```

The complete price list is returned in the *PriceListFile* XML file. Use your custom code to retrieve the **offer_price** objects contained in that price list.

Modifying Offer Profiles

> **Note:**
>
> If you introduce a new resource ID in an offer profile you modify, you must restart the CM after your load the offer profiles.

To modify the offer profile data in the BRM database:

1. Retrieve the offer profile data and use your custom code to retrieve the **offer_price** objects. See "Retrieving Offer Profiles".
2. Modify the **offer_price** objects as necessary.
3. Configure the offer profile data in XML format. See "Configuring Offer Profile Data".
4. Commit the modified offer profile data into the database. See "Storing Offer Profile Data".
5. If you added a new resource ID to any offer profile, stop and restart the CM.

Deleting Offer Profiles

Delete offer profiles using the **loadpricelist** utility in the following ways:

- Non-Interactive mode:

  ```
  loadpricelist -p
  ```

  This command deletes all the pricing objects from the database.

- Interactive mode:

  ```
  delete offer_profile PriceObject
  ```

> **Note:**
>
> If you use the **delete** command without any parameters, all in-memory pricing information is deleted from the database.

About Configuring Resources for Tracking by BRM

Configure the resources that BRM should track by setting up provisioning tags for your services as shown in the *BRM_Home*/**sys/data/config pin_offer_profile_provisioning_tags_policy_attributes.xml** file. For each provisioning tag, configure the types of resources valid for that tag and provide information on the specific

opcode which must be called to process changes to the product or discount associated with that provisioning tag.

The **pin_offer_profile_provisioning_tags_policy_attributes.xml** file contains the configuration for the default provisioning tag which BRM provides. See Example 20-1. The first section of this file lists the valid resources for the default provisioning tag. The next section contains information on the opcodes to be called when the product or discount containing the provisioning tag is purchased or canceled, the parameters that specify the fields to be added to the opcode's input flist, and the value for each field.

See "Configuring Provisioning Tags for Policy-Driven Charging" for more information.

Configuring Event Notification for Policy-Driven Charging

To configure event notification for policy-driven charging, you must merge the policy-driven charging event notification list with the general BRM event notification list.

Each event in an event notification list is mapped to the opcode or opcodes that are run when the event occurs. The event notification list for policy-driven charging contains of entries for offer profiles and for the products and discounts associated with them.

Example 18-1 shows the event notification list for offer profiles:

Example 18-1 Event Notifications Related to Offer Profiles

```
1301    0       /event/notification/offer_profile/create
1301    0       /event/notification/offer_profile/update
1301    0       /event/notification/offer_profile/delete
1301    0       /event/notification/offer_profile/ThresholdBreach
```

BRM uses the Enterprise Application Integration (EAI) framework publishing opcode, PCM_OP_PUBLISH_GEN_PAYLOAD (number 1301), to set up event notifications for offer profiles. This opcode is internal to BRM.

Example 18-2 shows the event notification list for products and discounts associated with offer profiles:

Example 18-2 Billing and Product Event Notification Entries

```
301     0       /event/billing/product/action/purchase
301     0       /event/billing/product/action/modify
301     0       /event/billing/product/action/cancel
301     0       /event/billing/product/action/modify/status
301     0       /event/billing/discount/action/purchase
301     0       /event/billing/discount/action/modify
301     0       /event/billing/discount/action/cancel
301     0       /event/billing/discount/action/modify/status
```

The **301** entry is the opcode number for the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode. See "About the PCM_OP_ACT_POL_EVENT_NOTIFY Policy Opcode".

For more information on event notification, see "Using Event Notification" in *BRM Developer's Guide*.

Merging the Policy-Driven Event Notification List with the BRM Event Notification List

To merge the policy-driven event notification list with the BRM event notification list:

1. Verify that the list entries for policy-driven charging are enabled. See "Verifying That Policy-Driven Charging Event Notification List Entries Are Enabled".
2. Merge the list for policy-driven charging with the general BRM list. See "Merging Event Notification Lists".
3. Load the updated event notification data into the BRM database. See "Loading the Updated Event Notification File into the BRM Database".

Verifying That Policy-Driven Charging Event Notification List Entries Are Enabled

To verify that the policy-driven charging event notification list entries are enabled:

1. Open the *BRM_Home*/**sys/data/config/pin_notify_ipc** file in a text editor.

> **Tip:**
>
> Save a copy of the default file before you change it.

2. Verify the following:
   - The offer profile notifications are enabled as shown in Example 18-1.
   - The billing event notifications for product and discount objects are enabled as shown in Example 18-2.
3. Save and close the file.

For more information, see "Editing the Event Notification List" in *BRM Developer's Guide*.

Merging Event Notification Lists

Merge the event notification list in the **pin_notify_ipc** file with the event notification list in the **pin_notify** file. For more information, see "Merging Event Notification Lists" in *BRM Developer's Guide*.

Loading the Updated Event Notification File into the BRM Database

To enable event notification for policy-driven charging, run the **load_pin_notify** utility to load the merged event notification list into the BRM database. For more information, see "Loading the Event Notification List" in *BRM Developer's Guide*.

Enabling Enterprise Applications to Handle Policy-Driven Charging Events

Complete the following operations to enable your enterprise applications to handle policy-driven charging event notifications that BRM publishes:

1. Verifying That BRM Is Integrated with Your Enterprise Applications
2. Enabling Enterprise Applications to Process Policy-Driven Charging Events
3. Specifying the Payload Configuration File to Use

For more information, see "Integrating BRM with Enterprise Applications" in *BRM Developer's Guide*.

Verifying That BRM Is Integrated with Your Enterprise Applications

For your enterprise applications to use the offer profile event notifications published by BRM, ensure that BRM is integrated with other applications in your enterprise. To do so, ensure that EAI Manager is installed.

For more information, see "About Installing EAI Manager" in *BRM Developer's Guide*.

Enabling Enterprise Applications to Process Policy-Driven Charging Events

You enable your enterprise applications to receive the notifications that BRM publishes when it processes policy-driven charging events.

When a customer buys, modifies, cancels a product or discount that is associated with an offer profile, BRM publishes a business event (such as **ProductPurchase** when the customer buys a product). If an offer profile is created, modified, deleted or there is an threshold breach, BRM publishes a related business event (for example, **OfferProfileCreate**, when an offer profile is initially stored in the database).

Business events associated with policy-driven charging are defined in the *BRM_Home***/sys/eai_js/payloadconfig_ipc.xml** payload configuration file. BRM provides default definitions of business events in the *BRM_Home***/sys/eai_js/payloadconfig.xml** payload configuration file.

If you already use a payload configuration file (for example, **payloadconfig.xml**) in your BRM environment, set up one payload configuration file to contain the definitions of all your business events.

Setting Up One Payload Configuration File

Set up one payload configuration file by doing the following:

1. Merge the contents of **payloadconfig_ipc.xml** (the payload configuration file for policy-driven charging) with your current payload configuration file (such as, **payloadconfig.xml**).

   > **Tip:**
   >
   > Save a copy of the default payload configuration files before merging them.

2. Save the merged file under a different name (for example, *your_payloadconfig.xml*).

Specifying the Payload Configuration File to Use

To specify the payload configuration file to use:

1. Open *BRM_home***/sys/eai_js/Infranet.properties** in a text editor.

   This is the payload generator external module **eai_js** properties file.

2. Point the **infranet.eai.configFile** entry to the appropriate payload configuration file.

   For example:

```
infranet.eai.configFile=/pinhome/pin201/opt/portal/12.0/sys/eai_js/your_payloadconfig.xml
```

   where *your_payloadconfig.xml* is the merged payload configuration file containing the business event entries associated with policy-driven charging.

3. Save and close the file.
4. Restart **eai_js**.

For more information, see "Configuring the EAI Payload for the Synchronization Queue DM" in *BRM Synchronization Queue Manager*.

Updating Reservation Preferences Configuration for Policy-Driven Charging

To update the reservation preferences configuration for services associated with policy-driven charging:

1. Add information on how the resource is to be rated (for example, by duration, by volume, by both duration and volume, or by occurrence). See "Updating Reservation Preferences Configuration Settings for Resources".

> **Note:**
>
> Policy-driven charging of services does not support prerated events.

2. Load the reservation preferences into the BRM database. See "Loading Reservation Preferences for Policy-Driven Charging".

Updating Reservation Preferences Configuration Settings for Resources

BRM stores the default entries for reservation preferences associated with policy-driven charging in the **pin_config_reservation_aaa_prefs_XXX** file. For example:

- **pin_config_reservation_aaa_prefs_gsm_telephony** (GSM telephony)
- **pin_config_reservation_aaa_prefs_gsm_data** (GSM data)
- **pin_config_reservation_aaa_prefs_gprs** (GPRS)

To configure the resource ID for the appropriate counters:

1. Open the *BRM_Home***/sys/data/config/pin_config_reservation_aaa_prefs_XXX** file in a text editor.

   For our example, open the *BRM_Home***/sys/data/config/pin_config_reservation_aaa_prefs_gsm_data** file, where *BRM_Home* is the directory in which BRM is installed.
2. Add an entry to specify the appropriate resource ID with the entity to be rated, such as duration (**2**), volume (**3**), duration and volume (**4**), or occurrence (**8**).

   For example:

   ```
   1 PIN_FLD_RESOURCE_ID INT [0] 1000009 in REQ_MODE 4
   ```

   Here, a noncurrency resource, (Megabytes Used) with the resource ID **100009** is associated with a volume-based request (**REQ_MODE 4**).
3. Save and close the file.

For more information, see "Editing the Event Notification List" in *BRM Developer's Guide*.

Loading Reservation Preferences for Policy-Driven Charging

To load the reservation preferences list for policy-driven charging:

1. Go to the *BRM_Home*/**sys/data/config** directory.
2. Use the following command to run the **load_config_reservation_aaa_prefs** utility:

   ```
   load_config_reservation_aaa_prefs -d -v load_config_reservation_aaa_prefs_XXX
   ```

   where:

   - `-d` creates a log file for debugging purposes.
   - `-v` displays information about successful or failed processing as the utility runs.
   - *load_config_reservation_aaa_prefs_XXX* is the reservation configuration file.

   For example:

   ```
   load_config_reservation_aaa_prefs -d -v pin_config_reservation_aaa_prefs_gsm_data
   ```

3. To verify that the reservation preferences were loaded, display the **/config/reserve** object by using Object Browser or the **robj** command with the **testnap** utility.
4. Stop and restart the CM.

Updating the Connection Manager for Policy-Driven Charging

If you use offer profiles in price lists, ensure that you update the CM configuration file, *BRM_Home*/**sys/cm/pin.conf**, for the following:

- Configure the memory required for offer profile cache. See "About Configuring the Required Memory for Offer Profiles Cache".
- Reference the system environment variables in the **pin.conf** file. See "About Referencing Offer Profile Related Variables in pin.conf Files".

About Configuring the Required Memory for Offer Profiles Cache

The offer profile cache called **fm_offer_profile_cache** is loaded with all **/offer_profile** objects at the initialization of the **fm_offer_profile** library. The **fm_offer_profile_cache** cache stores the last updated timestamp along with each **/offer_profile** object.

> **Note:**
>
> Restart the CM after you make any change to the offer profile configuration entry in the **pin.conf** file.

To update the **pin.conf** file for the **fm_offer_profile_cache** entry, see "Updating the pin.conf Configuration File for Offer Profiles".

About Referencing Offer Profile Related Variables in pin.conf Files

You can reference the system environment variable for offer profiles in **pin.conf** configuration file to facilitate future migration of the **pin.conf** file to BRM implementations on other platforms. See "System Administration pin.conf Reference" in *BRM System Administrator's Guide*.

> **Caution:**
>
> Restart the CM after you make any change to the **/offer_profile** object.

To update the **pin.conf** file for this entry, see "Updating the pin.conf Configuration File for Offer Profiles".

Updating the pin.conf Configuration File for Offer Profiles

To update the CM **pin.conf** file:

1. Open the *BRM_Home***/sys/cm/pin.conf** file in a text editor.)
2. Configure the required memory for the offer profile cache using appropriate values for *number_of_entries*, *cache_size*, *hash_size* in the following command. Add this line, if necessary.

   ```
   - cm_cache fm_offer_profile_cache number_of_entries cache_size hash_size
   ```

   For example:

   ```
   - cm_cache fm_offer_profile_cache 20, 102400, 13
   ```

   For more information, see "Improving Performance for Loading Large Product Offerings" in *BRM System Administrator's Guide*
3. Verify that the following command to reference the offer profile environmental variable from within the **pin.conf** file is present in the file:

   ```
   - cm fm_module ${PIN_HOME}/lib/fm_offer_profile${LIBRARYEXTENSION}
   fm_offer_profile_config fm_offer_profile_init pin
   ```

   Uncomment this command if it is commented, or add this command if it is not present in the **pin.conf** file.

   For more information, see "Preparing for Platform Migration by Using Variables in pin.conf Files" in *BRM System Administrator's Guide*.
4. Save the file.
5. Stop and restart the CM.

For more information, see "Improving Connection Manager Performance" in *BRM System Administrator's Guide*.

Customizing Information Received from BRM

BRM uses the following opcodes to support offer profile threshold notifications:

- PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS

- PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS
- PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD
- PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE
- PCM_OP_ACT_POL_EVENT_NOTIFY

This section describes these opcodes and how you can customize them, if necessary.

About the PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS Opcode

BRM uses the PCM_OP_BAL_ APPLY_MULTI_BAL_IMPACTS opcode to process both offline and online charging events. This opcode retrieves the current balance and calls PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode.

About the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS Policy Opcode

The PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode sets up an offer profile threshold breach notification when it encounters a threshold breach in a customer's usage of a resource defined in an offer profile (associated with the customer's purchased rate plan).

When the policy opcode is called by BRM as part of an authorization or reauthorization of a subscriber's request to use a service, it sets up an in-session notification for the offer profile threshold breach it encounters for the customer's usage of the resource. When the policy opcode is called by BRM during billing or accounts receivable operations, it sets up an out-of-session notification for the offer profile threshold breach it encounters for the customer's usage of the resource.

The PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode does not handle prerated events.

When you start (or restart) Connection Manager, BRM caches the set of unique resource IDs that you have configured in your offer profiles. The PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode uses this cache of resource IDs as the initial filter when applying the balance impact of each resource to retrieve the offer profile information and determine if an offer profile threshold notification is necessary. If a resource is not present in the cached entries, this policy opcode does not retrieve the offer profile information for that resource. Consequently, it will not set up an offer profile threshold breach notification, even if such a notification is necessary.

> **Note:**
>
> If you introduce a new resource ID, stop and restart the CM.

How Balances Are Processed for Out-of-Session Notifications

For out-of-session rating events, the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode uses the offer profile associated with the product or discount that granted this resource (called the grantor object).

The policy opcode retrieves all the services associated with the balance group and records an offer profile threshold breach notification event for all the services where an offer profile threshold was breached.

How Balances Are Processed for Notifications for Prepaid Sessions

For in-session rating events, the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode sets up offer profile threshold breach notifications for the subscriber only. It does not take resource sharing into consideration when it sets up these notifications.

This policy retrieves the offer profile associated with the purchased product or discount in the balance impact data it received from the rating process. It uses the offer profile information, the subscriber's current balances and the consumed reservation amount for the resource to determine whether an offer profile threshold was breached. If so, it records an offer profile threshold breach notification event.

Customizing Information in Offer Profile Threshold Breach Notifications

See "Sample Threshold Breach Notification" for the default information in an offer profile threshold breach notification. You can customize the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS policy opcode to retrieve other information on the resource usage.

About the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD Opcode

The PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode uses offer profiles passed in the input flist to retrieve the dynamic information for a given service identifier and resource ID.

BRM calls this opcode to perform the following operations:

- Determining Whether Balance Impacts Trigger Notifications
- Readjusting Quotas for Requests to Update and Authorize
- Readjusting Quotas for Other AAA Requests

For more information on the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode, see *Opcode Flist Reference*.

Determining Whether Balance Impacts Trigger Notifications

To determine whether any threshold breach notifications result from balance impacts due to the resource usages, BRM calls the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode with the balances (without current impact) stored in the BALANCES array of opcode's input flist.

The PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode does the following:

1. Calculates the sum of the current balance and the consumed reservation from the BALANCES array.
2. Locates old used quota and the last threshold notification.
3. Calls the PCM_OP_BAL_GET_PREPAID_BALANCES opcode with the resource ID to compute the new used amount.

The PCM_OP_BAL_GET_PREPAID_BALANCES opcode returns the sum of the updated current balance and consumed reservation as the new used amount.

4. Locates the status label, offer profile name and policy label name for the tier with the nearest threshold. To do so, it employs the new used amount and the range from the last threshold notification.
5. Calculates the offset to the next threshold as the difference between the new used amount and the next threshold on that tier.
6. Sets the PIN_FLD_INDICATOR field in the output to **0** if there is a breach. And returns the calculated offset to the next threshold.

Readjusting Quotas for Requests to Update and Authorize

BRM calls the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode with the additional quota reported by the network in the QUANTITY field of the input list.

The PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode does the following:

1. Calls the PCM_OP_BAL_GET_PREPAID_BALANCES opcode with the resource ID to retrieve the current balance and consumed reservation.
2. Calculates the total used as the sum of the current balance, the consumed reservation, and the additional quantity reported by the network.
3. Locates the status label, offer profile name, and policy label name for the tier with the nearest threshold with reference to the total used amount.
4. Calculates the offset to the next threshold as the difference between the new used amount and the next threshold on that tier.
5. Returns the calculated offset to the next threshold

Readjusting Quotas for Other AAA Requests

When called to readjust the quota for AAA requests other than the update and reauthorize request, the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode does the following:

1. Retrieves the current balance and consumed reservation by calling the PCM_OP_BAL_GET_PREPAID_BALANCES opcode with the resource ID.
2. Calculates the total used as the sum of the current balance and the consumed reservation.
3. Locates the status label, offer profile name and policy label name for the tier with the nearest threshold with reference to the total used amount.
4. Calculates the offset to the next threshold as the difference between the new used amount and the next threshold on that tier.
5. Returns the calculated offset to the next threshold.

About the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE Opcode

The PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode returns the following information from the database:

- All the information associated with a subscriber's mobile station ID (MSID)
- Specific service only for a given object identifier (POID)

The PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode uses the internal PCM_OP_OFFER_PROFILE_GET_OFFER_PROFILE opcode to fetch the necessary information. It returns the information in the PIN_FLD_RESULTS array with PIN_FLD_ACCOUNT containing information associated with the account and PIN_FLD_SERVICE_INFO containing information on the services.

The opcode returns the following information based on the value you input for the PIN_FLD_MODE entry when you call this opcode:

- Static information on the resource used by a subscriber such as offer profiles and subscriber's preference data (PIN_FLD_MODE is **1**)
- Dynamic information such as current balances for the various resources (PIN_FLD_MODE is **2**)
- Both types of information (PIN_FLD_MODE is **3**)

For more information on the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode, see *Opcode Flist Reference*.

Customizing Information Received from BRM

Use the PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE policy opcode to customize the information your network policy controlling application retrieves from BRM.

By default, PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE policy opcode calls the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE opcode and returns the output from that opcode.

For more information on the PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE policy opcode, see *Opcode Flist Reference*.

About the PCM_OP_ACT_POL_EVENT_NOTIFY Policy Opcode

When a product or discount associated with an **offer_profile** is purchased, canceled, or modified, the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode calls the PCM_OP_OFFER_PROFILE_GET_OFFER_PROFILE opcode to retrieve the associated offer profile data and adds it to the event notification. The PCM_OP_OFFER_PROFILE_GET_OFFER_PROFILE opcode is internal to BRM. See "About the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE Opcode" for more information.

For more information on the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode, see *Opcode Flist Reference*.

Customizing Information in Event Notifications for Policy-Driven Charging

Use the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode to customize the event notification information associated with policy-driven charging that you want to receive from BRM.

Managing Offer Profiles with Opcodes

You can manage offer profile data using price list opcodes and offer profile opcodes.

Managing Offer Profile Data with Price List Opcodes

Use the PCM_OP_PRICE_SET_PRICE_LIST and PCM_OP_PRICE_GET_PRICE_LIST opcodes to manage your offer profiles.

Creating, Modifying, and Deleting Offer Profiles with PCM_OP_PRICE_SET_PRICE_LIST

The PCM_OP_PRICE_SET_PRICE_LIST opcode searches the database for each offer profile in the PIN_FLD_OFFER_PROFILES array. To create or modify offer profiles in the database, input the **/offer_profile** objects in the PIN_FLD_OFFER_PROFILES array. To delete an offer profile from the database, input the **/offer_profile** object in the PIN_FLD_OFFER_PROFILES array and specify the PIN_FLD_DELETE_FLAG entry in the input flist.

The PCM_OP_PRICE_SET_PRICE_LIST opcode searches the database for each offer profile in the PIN_FLD_OFFER_PROFILES array and does one of the following:

- If the offer profile is not found, the opcode creates and stores an **/offer_profile** object with this data.
- If the offer profile is found, the opcode retrieves that **/offer_profile** object and does one of the following:
  - If the PIN_FLD_DELETE_FLAG entry is not present in the input flist, the opcode updates **/offer_profile**, and stores the updated object.
  - If the PIN_FLD_DELETE_FLAG entry is present in the input flist, the opcode deletes the **/offer_profile** object from the database.

See "Managing Price Lists" for more information.

Retrieving Offer Profiles with PCM_OP_PRICE_GET_PRICE_LIST

Use the PCM_OP_PRICE_GET_PRICE_LIST opcode to retrieve the contents of an **/offer_profile** object from the database. The PIN_FLD_OFFER_PROFILES array in the output contains the **/offer_profile** objects.

See "Managing Price Lists" for more information.

Managing Offer Profiles with Offer Profile Opcodes

Manage offer profile data using the following:

- PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE. See "About the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE Opcode".
- PCM_OP_TCF_AAA_POL_GET_SUBSCRIBER_PROFILE. See "About the PCM_OP_TCF_AAA_GET_SUBSCRIBER_PROFILE Opcode".

About the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE Opcode

Use the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE opcode to create, modify, and delete offer profiles.

Input the offer profiles in the PIN_FLD_OFFER_PROFILES array. To delete an offer profile, specify the PIN_FLD_DELETE_FLAG entry for the offer profile in the input flist.

The opcode searches the database for the offer profiles in the PIN_FLD_OFFER_PROFILES array and does one of the following:

- If the offer profile is not found, the opcode creates and stores an **/offer_profile** object with this data.
- If the offer profile is found, the opcode retrieves that **/offer_profile** object and
  - If the PIN_FLD_DELETE_FLAG entry is not present in the input flist, it updates **/offer_profile**, and stores the updated object.
  - If the PIN_FLD_DELETE_FLAG entry is present in the input flist, it deletes the **/offer_profile** object.

For more information on the PCM_OP_OFFER_PROFILE_SET_OFFER_PROFILE opcode, see *BRM Developer's Reference*.

Customizing Subscriber Preferences Data for Policy-Driven Charging

BRM stores the default entries for subscriber preferences associated with policy-driven charging in the **config_subscriber_preferences_map.xml** file.

To customize subscriber preferences data for policy-driven charging:

1. Open the *BRM_Home***/sys/data/config/config_subscriber_preferences_map.xml** configuration file in a text editor.
2. Edit the file as necessary. You can add other subscriber preferences.

   The elements in the XML file are described in the discussion on subscriber preferences in *BRM Telco Integration*.
3. Save and close the file.

Customer Center uses the configurations in the **/config/subscriber_preferences_map** object to dynamically list the preferences that a subscriber can configure. You can customize the information as necessary.

Customizing Business Events Associated with Policy-Driven Charging

If you customize business events associated with policy-driven charging, include your definitions of these events in the **payloadconfig_ipc.xml** configuration file. For more information, see "About Publishing Additional Business Events" in *BRM Developer's Guide*.

Sample Threshold Breach Notification

Example 18-3 shows the contents of a sample offer profile threshold breach notification.

Example 18-3 Sample Offer Profile Threshold Breach Notification

```
PIN_EVENT_TY('ThresholdBreach', '<?xml version="1.0"?>
<ThresholdBreach Version="1.0">
   <service_obj>0.0.0.1 /service/telco 223789 7</service_obj>
   <account_obj>0.0.0.1 /account 10 1 </account_obj>
   <login>test</login>
   <alias_list>
```

```
      <name>694-20120607-231907-3-9178 --157566752-slc.example.com</name>
   </alias_list>
   <resource_list>
      <resource_name>Megabytes Used</resource_name>
      <resource_id>100009</resource_id>
      <current_bal>35.28</current_bal>
      <unit>0</unit>
      <threshold_breach >
     <status_label>HIGH_QOS</status_label>
     <delta_to_next_threshold>14.72</delta_to_next_threshold>
     <policy_label>Fair Usage</policy_label>
     <offer_profile_name>platinum</offer_profile_name>
     </threshold_breach>
   </resource_list>
</ThresholdBreach>
```

19

Working with Promotions

This chapter describes how to implement promotions in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Promotions

You can apply promotions to your customers, such as by granting 100 free minutes, 2 GB of free data, or 2 free movie downloads, in the following situations:

- On special dates, such as a customer's birthday or membership anniversary. See "Applying Promotions On Special Dates".
- Events or actions that occur in BRM, such as a customer purchasing a product or successfully topping up an account balance. See "Applying Promotions for Special Events or Actions".

Applying Promotions On Special Dates

You specify whether customers qualify for a promotion based on a special date by adding a promotion tag and date to their account profile. For example, to apply a promotion on a customer's birthday, you could add the promotion tag BIRTHDAY and promotion date of June 15. On June 15, BRM could grant the customer 100 free minutes, 5 Gigabytes of free data, or so on.

BRM applies the promotions to your customers' accounts when the **pin_apply_promotion** utility is run. The utility searches through each account's profile for a promotion date that occurs on or before the current date. When it finds a match, it looks up the promotion event associated with the promotion tag and then rates the promotion event, which in turn grants the free resources.

To configure BRM to apply promotions on special dates:

1. When you create an account or modify an account profile, include a promotion tag and date.

   The promotion data is stored in an account's **/profile/specialdates** object.

2. Create a promotion event (**/event/promotion/** subclass) for each promotion that your company supports and then load the promotion events into the BRM database.

   For example, you could create an **/event/promotion/birthday** storable class for applying birthday promotions. BRM uses this event for rating and applying the promotion to your customers' accounts.

   For more information about:

- Creating custom events, see "Creating Service and Event Storable Classes" in *BRM Developer's Guide*.
- Loading custom events into the BRM database, see "load_event_map".

3. Map your promotion tags to the promotion events you created in step 2. See "Mapping Promotion Tags to Promotion Events".

   The mappings are stored in the **/config/event_promo_tag_map** object.

4. Define the amount and type of free resources that BRM needs to grant your customers when rating each promotion event. To do so, define how to apply credits for each promotion event in your system-level or account-level charge offers and rates.
5. Optionally, customize how promotions are applied. See "Customizing Special Date, Event, or Action Promotions" in *BRM Opcode Guide*.
6. Run the **pin_apply_promotion** utility on a daily basis. See "pin_apply_promotion".

Mapping Promotion Tags to Promotion Events

To map promotion tags to promotion events:

1. Open the *BRM_home***/sys/data/config/config_event_promo_tag_map.xml** file in a text editor.

   The **config_event_promo_tag_map.xml** file contains sample mappings, but you can modify them to meet your business needs.

2. Add an **<EVENT_MAP>** array element for each promotion tag that you want mapped to a promotion event. Table 19-1 describes the elements in **<EVENT_MAP>**.

Table 19-1 Elements in config_event_promo_tag_map.xml

Element	Description
<PROMO_EVENT_TYPE>	The promotion event to rate.
<PROMO_TAG>	The name of the promotion tag.

The following sample content maps the **BIRTHDAY** promotion tag to the **/event/promotion/birthday** promotion event:

```
<ConfigObject>
   <DESCR>Event Promo Tag Map Configuration</DESCR>
   <HOSTNAME>-</HOSTNAME>
   <NAME>Event Promo Tag Map</NAME>
   <PROGRAM_NAME>load_config</PROGRAM_NAME>
   <EVENT_MAP elem="0">
      <PROMO_EVENT_TYPE>/event/promotion/birthday</PROMO_EVENT_TYPE>
      <PROMO_TAG>BIRTHDAY</PROMO_TAG>
   </EVENT_MAP>
</ConfigObject>
```

3. Save and close the file.
4. Edit the **load_config** utility's configuration file (*BRM_home***/apps/load_config/pin.conf**) to specify how to connect to the BRM database.

5. Load the **config_event_promo_map.xml** file into the BRM database:

   ```
   load_config -d -v BRM_home/sys/data/config/config_event_promo_tag_map.xml
   ```

   See "load_config" in *BRM Developer's Guide* for more information.

6. Stop and restart the Connection Manager (CM).

7. Verify that the mappings were loaded properly by running the following command:

   ```
   load_config -w "event_promo_tag_map" config_event_promo_tag_map_out.xml
   ```

Applying Promotions for Special Events or Actions

You can configure BRM to apply promotions for special events or actions by associating BRM event types with your promotions. For example, you could specify that all **/event/billing/product/action/purchase** events qualify for a promotion. In this case, every time customers purchase a new product, BRM could grant them 5 gigabytes of free data, 1 free movie download, or so on.

You implement these types of promotions by using the BRM event notification system. When a specified BRM event occurs, the event notification system triggers a call to the PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT opcode which in turn looks up the promotion event associated with the BRM event. For example, it could look up that **/event/billing/product/action/purchase** events are associated with the **/event/promotion/subscribe_offer** promotion event. BRM then rates the promotion event, which grants the free resources.

To configure BRM to apply promotions for special events or actions:

1. Create a promotion event (**/event/promotion/** subclass) for each promotion that your company supports and then load the promotion events into the BRM database.

   For example, you could create an **/event/promotion/subscribe_offer** storable class. BRM uses this event for rating and applying the promotion to your customers' accounts.

   For more information about:

   - Creating custom events, see "Creating Service and Event Storable Classes" in *BRM Developer's Guide*.
   - Loading custom events into the BRM database, see "load_event_map".

2. Configure the event notification system to trigger calls to the PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT opcode when a BRM event occurs. See "Configuring Event Notification for Promotions".

3. Map BRM events to the promotion events you created in step 1. See "Mapping BRM Events to Promotion Events".

   The mappings are stored in the **/config/event_promo_map** object.

4. Define the amount and type of free resources that BRM needs to grant your customers when rating each promotion event. To do so, define how to apply credits for each promotion event in your system-level or account-level charge offers and rates.

5. Optionally, customize how promotions are applied. See "Customizing Special Date, Event, or Action Promotions" in *BRM Opcode Guide*.

Configuring Event Notification for Promotions

To configure event notification for promotions, you must map each BRM event to which you want to apply promotions to the PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT opcode. For more information about event notification, see "Using Event Notification" in *BRM Developer's Guide*.

To configure event notification for promotions:

1. Open the *BRM_home***/sys/data/config/pin_notify** file in a text editor.
2. Add an entry for each BRM event that should trigger calls to opcode **9075** (the number for the PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT opcode).

   For example, to trigger calls to PCM_OP_SUBSCRIPTION_HANDLE_PROMO_EVENT whenever customers successfully top up their account balances, purchase a new product, or create an account profile, you would add the following lines:

   ```
   # Event Notification for Promotions
   9075    0       /event/billing/topup
   9075    0       /event/billing/product/action/purchase
   9075    0       /event/notification/profile/create
   ```

3. Save and close the file.
4. Load your **pin_notify** file into the BRM database by using the **load_pin_notify** utility:

   load_pin_notify *BRM_home*/**sys/data/config/pin_notify**

   See "load_pin_notify" in *BRM Managing Customers* for more information.
5. Stop and restart the Connection Manager (CM).

Mapping BRM Events to Promotion Events

To map BRM events to promotion events:

1. Open the *BRM_home***/sys/data/config/config_event_promo_map.xml** file in a text editor.

   The **config_event_promo_map.xml** file contains sample mappings, but you can modify them to meet your business needs.
2. Add an **<EVENT_MAP>** array element for each BRM event that you want mapped to a promotion event. Table 19-2 describes the elements in **<EVENT_MAP>**.

Table 19-2 Elements in config_event_promo_map.xml

Element	Description
<EVENT_TYPE>	The BRM event.
<FIELD_NUM>	The field number defined in the *BRM_home***/include/pin_flds.h** file. For example, you would use field number **601** for PIN_FLD_LABEL. **Note**: This applies to fields with a data type of PIN_FLDT_STR only.
<FIELD_VALUE>	The value of the field specified in FIELD_NUM.
<PROMO_EVENT_TYPE>	The promotion event.

The following sample content maps the **/event/billing/topup** event to the **/event/promotion/topup** promotion event:

```
<ConfigObject>
   <DESCR>Event Promo Map Configuration</DESCR>
   <HOSTNAME>-</HOSTNAME>
   <NAME>Event Promo Map</NAME>
   <PROGRAM_NAME>load_config</PROGRAM_NAME>
   <EVENT_MAP elem="0">
      <EVENT_TYPE>/event/billing/topup</EVENT_TYPE>
      <FIELD_NUM>55</FIELD_NUM>
      <FIELD_VALUE>Topup Event (acct)</FIELD_VALUE>
      <PROMO_EVENT_TYPE>/event/promotion/topup</PROMO_EVENT_TYPE>
   </EVENT_MAP>
</ConfigObject>
```

3. Save and close the file.
4. Edit the **load_config** utility's configuration file (*BRM_home***/apps/load_config/pin.conf**) to specify how to connect to the BRM database.
5. Load the **config_event_promo_map.xml** file into the BRM database:

   load_config -d -v *BRM_home***/sys/data/config/config_event_promo_map.xml**

   See "load_config" in *BRM Developer's Guide* for more information.
6. Stop and restart the Connection Manager (CM).
7. Verify that the mappings were loaded properly by running the following command:

   load_config -w "event_promo_map" config_event_promo_map_out.xml

20

Working with Provisioning Tags

This chapter describes how to implement Oracle Communications Billing and Revenue Management (BRM) provisioning tags if you are using Pricing Center.

Topics in this document:

About Provisioning Tags

Provisioning tags implement extended rating attributes (ERAs) or other user-defined attributes that can enable rating or discounting to vary for a service. For telco services, you can also use provisioning tags to provision supplementary services and service extensions.

An offer profile and the provisioning tag for the associated product or discount use the same name and that name must be unique. If you create an offer profile to associate with an existing product or discount, use the provisioning tag to name the offer profile. If you configure a new product or discount around an existing offer profile, use the appropriate offer profile name as the provisioning tag for the product or discount.

You can use one of the following methods to define provisioning tags, depending on what the tag is for:

- **Provisioning tag framework**: You create provisioning tags by defining them in an XML file that is loaded into a configuration object in BRM. You can create provisioning tags for any service or for an account.

  You also might need to add the tag to the provisioning policy source file and compile the file.

  See "Using the Provisioning Tag Framework".

  > **Note:**
  >
  > – For telco services, use Services Framework Manager to define provisioning tags in most cases. You must use Services Framework Manager if the provisioning tag creates supplementary services or service extensions. See "About Provisioning Tags for Telco Services " in *BRM Provisioning Services*.
  >
  > – For non-telco services, the provisioning tag framework is the recommended method of creating provisioning tags.

- **Provisioning Tags application in Pricing Center**: For telco services only, you can create service-level provisioning tags with this application. You can include service-level ERAs, supplementary services, and service extensions in the tags.

> **Note:**
>
> You can create provisioning tags that include existing ERAs only.

- **Telco tag text file**: For telco services only, you can create service-level provisioning tags by defining them in a text file and then loading the file into BRM. You can include supplementary services and ERAs in the tags.

  You can also create account-level ERAs in this same file.

  See "Defining Provisioning Tags for Telco Services by Using the pin_telco_tags File " in *BRM Provisioning Services*.

- **Provisioning policy source file**: You can define product provisioning tags by editing and compiling the policy file that is the source file for all provisioning operations.

> **Note:**
>
> The provisioning tag framework is the preferable method for creating provisioning tags.

See "Using a Policy Source File to Set Up Provisioning".

Defining Provisioning Tags for Telco Services by Using the pin_telco_tags file

This section describes defining provisioning tags through the **pin_telco_tags** file for a specific telco service. For example, you use **pin_telco_tags_gsm** for GSM provisioning tags.

You can include service-level ERAs, supplementary services, and service extensions in a provisioning tag defined in a **pin_telco_tags** file.

You can use the Provisioning Tags application in Pricing Center instead of the **pin_telco_tags** file to define provisioning tags for telco services. But you cannot create custom ERAs using Provisioning Tags. For information, see Provisioning Tags Help.

To define provisioning tags using the **pin_telco_tags** file:

- Configure provisioning tags in the **pin_telco_tags**_*service* file. See "Configuring Provisioning Tags in the pin_telco_tags File".
- Load the **pin_telco_tags**_*service* file into the BRM database with the **load_pin_telco_tags** utility. See "Loading the pin_telco_tags File ".

You can also define account-level ERAs in the **pin_telco_tags** file. See "Defining Account-Level ERAs in the pin_telco_tags File ".

Configuring Provisioning Tags in the pin_telco_tags File

To configure a provisioning tag in the **pin_telco_tags** file:

1. Open the **pin_telco_tags**_*service* file. For example, use **pin_telco_tags_gsm** for GSM services.

The default **pin_telco_tags** files are in *BRM_Home*/**sys/data/config**. They include examples and instructions.

2. Use this syntax to add a provisioning tag:

```
provisioning_tag  "Class" "ProvTag" "PTDescription" "DelayedProvReqd"
service_extn      "Extension Type Name" "Extension Value"
features          "One Or More Feature Name String Values"
service_era       "ServiceERA" "StringIdERA" "StringIdServiceERADesc" "ProvBool" "ERALabel"
```

Enter each value in quotation marks.

A provisioning tag can be any combination of service extensions, features, and service-level ERAs. You do not need to include all three types of data in a tag.

Table 20-1 describes the provisioning tag syntax:

Table 20-1 Provisioning Tag Syntax

Tag Element	Value	Description
provisioning_tag	Class	The object that stores the tag. For example: **"/config/telco/gsm/telephony"**
provisioning_tag	ProvTag	The name of the provisioning tag. For example: **"DataPremium"**
provisioning_tag	PTDescription	The description of the provisioning tag. For example: **"Premium Data Service"**
provisioning_tag	DelayedProvReqd	Whether the tag is unprovisioned when the product containing the tag is canceled. The possible values are: • **"y"** specifies that cancellation triggers unprovisioning. In most cases, use this setting. • **"n"** specifies that cancellation does not trigger unprovisioning. Use this setting to leave a customer's service configuration unchanged. For example, you might want to leave a voice mailbox intact.
service_extn	Extension Type Name	The type of service extension. For example: **"BEARER_SERVICE"**
service_extn	Extension Value	The code for a GSM bearer service or other service extension. For example: **"B46"** Codes are defined in the GSM specification. You must use the exact code that the network requires.
features	One or More Feature Name String Values	The GSM supplementary services that are provisioned when this product is purchased. The services are entered as codes, in one line. For example: **"CLIP" "CW"** These codes are defined in the GSM specification. You must use the exact code that the network requires.
service_era	ServiceERA	The service ERA code. For example: **"FRIENDS_FAMILY"**
service_era	StringIdERA StringIdServiceERADesc	The IDs for the ERA name and description. For example: **"12" "13"** You define a localized name and description for these IDs in the **era_descr.***locale*file. These names and descriptions appear in Customer Center.

Table 20-1 (Cont.) Provisioning Tag Syntax

Tag Element	Value	Description
service_era	ProvBool	Whether or not provisioning is required. The possible values are: • **"y"** specifies that provisioning is required. • **"n"** specifies that provisioning is not required.
service_era	ERALabel	The name of a list within the ERA. An ERA can have one or more lists. For example: **"MYFRIENDS"** **Note:** You cannot localize the ERA label. You cannot have duplicate label names associated with the same ERA code.

This example shows a provisioning tag for a telephony product that includes a bearer service, call waiting and voice mailbox supplementary services, and friends and family service-level ERAs:

```
# Standard Telephony Package
provisioning_tag  "/config/telco/gsm/telephony"  "Voice Standard" "Voice Standard
Service package with Called ID, Call Waiting, Voice mail and Friends and Family" "y"
service_extn      "BEARER_SERVICE" "T11"
features          "CLIP" "CW" "VMBOX"
service_era       "FRIENDS_FAMILY" 12 13 "n" "MYFRIENDS"
service_era       "FRIENDS_FAMILY" 12 13 "n" "MYFAMILY"
```

Loading the pin_telco_tags File

Run the **load_pin_telco_tags** utility to load the contents of the **pin_telco_tags**_*service* file: for example, the **pin_telco_tags_gsm** file: into the BRM database. This utility creates or updates **/config/telco/***service* and **/config/account_era** objects.

> **Note:**
>
> By default, the **load_pin_telco_tags** utility appends telco provisioning tags and account-level ERAs to the BRM database. But if you use the **-x** parameter, this utility overwrites existing telco provisioning tags and account-level ERAs. Do not use the **-x** parameter unless you are certain you want to overwrite existing objects.

> **Note:**
>
> The **load_pin_telco_tags** utility requires a configuration file. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

> **Note:**
>
> You cannot create ERAs for individual brands. All ERAs can be used by any brand.

1. Edit the **pin_telco_tags**_*service* file to add the custom account-level ERAs. The default **pin_telco_tags_gsm** file in *BRM_Home***/sys/data/config** includes examples and instructions.
2. Save the **pin_telco_tags**_*service* file.
3. Use the following command to run the **load_pin_telco_tags** utility:

   ```
   load_pin_telco_tags pin_telco_tags_service
   ```

4. Restart the Connection Manager (CM).
5. Restart Pricing Center.

To verify that the account ERAs were loaded, you can display the **/config** objects by using the Object Browser or use the **robj** command with the **testnap** utility.

Defining Account-Level ERAs in the pin_telco_tags File

Account-level ERAs apply to any activity in an account no matter which service is involved.

You define account-level ERAs in any **pin_telco_tags**_*service* file. You can use the define account-level ERAs in the same file you used for telco provisioning tags.

> **Note:**
>
> You cannot use the Provisioning Tags application for account-level ERAs.

To define an account-level ERA:

1. Open a **pin_telco_tags**_*service* file. You can define an account-level ERA in any **pin_telco_tags** file, because they are not associated with a particular service.

   The default **pin_telco_tags** files are in *BRM_Home***/sys/data/config**. They include examples and instructions.

   Account-level ERAs are added in a block at the end of the file.
2. Use this syntax to add an account-level ERA:

   ```
   account_era  "AccountERA" "StringIdERA" "StringIdAccountERADesc"
   ```

Table 20-2 describes the account ERA syntax:

Table 20-2 Account ERA Syntax

Value	Description
AccountERA	The account ERA code. For example: **"SPECIAL_DAY"**
StringIdERA StringIdAccountERADesc	The IDs for the ERA name and description. For example: **"2" "3"** You define a localized name and description for these IDs in the **era_descr.***locale*file. These names and descriptions appear in Customer Center. See "Localizing and Customizing Strings" in *BRM Developer's Guide* for information about updating this file.

For example:

```
account_era "SPECIAL_DAY" "2" "3"
```

Deciding Which Provisioning Tag Method to Use

For telco services, you should create provisioning tags by using one of these methods. The order of preference is shown below.

1. Provisioning tag framework. See "Using the Provisioning Tag Framework".
2. Provisioning Tags application in Pricing Center.
3. Telco tag text file.
4. Provisioning policy source file. See "Using a Policy Source File to Set Up Provisioning".

For non-telco services, you should create provisioning tags by using one of these methods. The order of preference is shown below:

1. Provisioning tag framework. See "Using the Provisioning Tag Framework".
2. Provisioning policy source file. See "Using a Policy Source File to Set Up Provisioning".

Using the Provisioning Tag Framework

You can create service-level or account-level provisioning tags using the provisioning tag framework. This framework stores provisioning tags in the **/config/provisioning_tag** object.

> **Note:**
>
> To create most provisioning tags for telco services, use the Provisioning Tags application in Pricing Center or the **load_pin_telco_tags** utility.

You define a provisioning tag by specifying the services to which it applies and the opcodes to run when the product or discount containing the tag is purchased or canceled. You might need to create custom opcodes for some provisioning tags.

To create a provisioning tag using this framework, do the following:

- Configure the provisioning tag in the **pin_config_provisioning_tags.xml** file. See "Configuring Provisioning Tags ".
- Load the provisioning tag configuration into the **/config/provisioning_tag** object with the **load_config_provisioning_tags** utility. "Loading Provisioning Tag Configurations".
- Add the tag to the provisioning policy source file and compile the file. You do this if you will include the provisioning tag in a product and if the tag uses a service associated with the __DEFAULT__ provisioning tag. See "Modifying and Compiling the Provisioning Policy Source File".

Configuring Provisioning Tags

To create provisioning tags, you configure the provisioning tags configuration file, **pin_config_provisioning_tags.xml**. This file is located in the *BRM_Home*/**sys/data/config** directory.

To define a provisioning tag in this configuration file, you specify the following:

- A unique name for the provisioning tag.

> **Note:**
>
> The provisioning tag for a product or discount must be the name of the offer profile with which the product or discount is associated.

- For a service-level tag, the permitted services.
- For an account-level tag, **/account**.
- The name and number of each opcode to run when the product or discount containing the provisioning tag is purchased or canceled. These opcodes contain the business logic to perform the actual provisioning, such as creating a profile.
- Parameters that specify the fields to be added to each opcode's input flist and the value for each field.

> **Note:**
>
> – Do not remove existing provisioning tags from **pin_config_provisioning_tags.xml** when adding new tags unless you want existing tags removed from the **/config/provisioning_tag** objects in the database.
>
> – When you load **pin_config_provisioning_tags.xml**, all existing instances of **/config/provisioning_tag** are removed from the database, so only the provisioning tags defined in the file when you load it will be in the database.

Table 20-3 lists the elements in the **pin_config_provisioning_tags.xml** file, the syntax to use for each element, and a description of how to specify each element. The syntax is based on the default version of the file:

Table 20-3 Elements in pin_config_provisioning_tags XML File

Element	Syntax	Description
BusinessConfiguration	`<BusinessConfiguration xmlns="http://www.portal.com/schemas/BusinessConfig" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig business_configuration.xsd">`	The root element common to all BRM configurations.
ProvisioningTag Configuration	`<ProvisioningTagConfiguration>`	Opens the provisioning tag configuration. Contains the ProvisioningTagList element.
ProvisioningTagList	`<ProvisioningTagList>`	Contains all the provisioning tag definitions.

Table 20-3 (Cont.) Elements in pin_config_provisioning_tags XML File

Element	Syntax	Description
ProvisioningTag	`<ProvisioningTag name= "__DEFAULT__">`	Contains the definition of a provisioning tag and specifies the tag's name.
PermittedTypes	`<PermittedTypes>/service/ip </PermittedTypes>`	Specifies a service or account that is valid for the provisioning tag. One or more of these tags can be part of a provisioning tag definition. When the API for getting a list of provisioning tags is called, the array of permitted types is looked up and only provisioning tags applicable to the specific service type are returned.
OpcodeList	`<OpcodeList>`	Contains the definition of one opcode. The opcodes specified for a provisioning tag contain the business logic required for provisioning when purchasing and canceling a product. A provisioning tag can include multiple opcodes.
OpcodeName	`<OpcodeName> PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD _PROVISIONING </OpcodeName>`	Specifies the opcode's name. See "Specifying an Opcode in a Provisioning Tag to Create an ERA".
OpcodeNumber	`<OpcodeNumber>417 </OpcodeNumber>`	Specifies the hard-coded number for an opcode. To find an opcode's number, see the opcode header files in the *BRM_Home*/**include/ops** directory.
OpcodeMode	`<OpcodeMode>0</OpcodeMode>`	Indicates when the opcode should be called, as follows: • **0**: on product or discount purchase • **1**: on product or discount cancellation • **2**: on both purchase and cancellation
OpcodeParamsList	`<OpcodeParamsList>`	Defines a parameter for an opcode. You can have multiple parameters. Each parameter is a name-value pair.

Table 20-3 (Cont.) Elements in pin_config_provisioning_tags XML File

Element	Syntax	Description
OpcodeParamName	`<OpcodeParamName>PIN_FLD_POID</OpcodeParamName>`	Specifies a field name to be added to the input flist. If the field is part of a substruct or array, use a period to separate the substruct or array name and the field name. For example: `PIN_FLD_EVENT.PIN_FLD_ACCOUNT_OBJ`
OpcodeParamValue	`<OpcodeParamValue>$SERVICE$</OpcodeParamValue>`	Specifies the value of the field. For certain values that are not known until run time, you can use a variable, such as $SERVICE$. See "Variables for Parameter Values".

For an example of the XML syntax, see "Sample Provisioning Tag XML File".

Specifying an Opcode in a Provisioning Tag to Create an ERA

When defining a configuration for the provisioning tag framework, you specify one or more opcodes to perform an action, such as creating an ERA. You can specify that an opcode run when a product or discount is purchased, canceled, or both. You can use an existing opcode or design a custom opcode.

If the provisioning tag is designed to create an ERA, you can specify that the PCM_OP_SUBSCRIPTION_PROVISION_ERA opcode run at both purchase and cancellation time. This opcode creates, modifies, or deletes a profile (**/profile** object). ERAs are stored in profiles.

The actions PCM_OP_SUBSCRIPTION_PROVISION_ERA takes depends on the value specified for the PIN_FLD_ACTION parameter in the provisioning tag:

- If PIN_FLD_ACTION is **Purchase**, the opcode checks if a profile already exists. If the profile does not exist, the opcode calls PCM_OP_CUST_CREATE_PROFILE to create the profile. If the profile does exist, the opcode calls PCM_OP_CUST_MODIFY_PROFILE to add data passed in from the input flist and to increment the reference counter by 1.
- If PIN_FLD_ACTION is **Cancel**, the opcode decrements the reference counter by 1. If the counter is 0, the opcode calls PCM_OP_CUST_DELETE_PROFILE to delete the profile.

For example, you can create a friends and family ERA for a service by calling the PCM_OP_SUBSCRIPTION_PROVISION_ERA at purchase time with the parameters in Table 20-4:

Table 20-4 PCM_OP_SUBSCRIPTION_PROVISION_ERA Parameters

OpcodeParamName	OpcodeParamValue
PIN_FLD_POID	0.0.0.0 /profile/serv_extrating -1
PIN_FLD_ACCOUNT_OBJ	$ACCOUNT$
PIN_FLD_FLAGS	0

Table 20-4 (Cont.) PCM_OP_SUBSCRIPTION_PROVISION_ERA Parameters

OpcodeParamName	OpcodeParamValue
PIN_FLD_SERVICE_OBJ	$SERVICE$
PIN_FLD_STR_VAL	12, 13
PIN_FLD_NAME	FRIENDS_FAMILY
PIN_FLD_INHERITED_INFO. PIN_FLD_EXTRATING. PIN_FLD_LABEL	MYFRIENDS

These name-value pairs indicate that an ERA named FRIENDS_FAMILY and an ERA label named MYFRIENDS are created. Because both the account and service POIDs are specified, the opcode creates a service-level profile (**/profile/serv_extrating**) object. If the service POID was not specified, the opcode would create an account-level profile (**/profile/acct_extrating**) object.

Note:

- PIN_FLD_POID is the POID, in string format, of the object the provisioning tag will create. This is converted to a POID when the opcode runs. If you are using multiple database schemas, the string is converted to the correct schema database number.
- For a service-level profile, the POID type is **/profile/serv_extrating**, as in the previous example. For an account-level profile, the POID type is **/profile/acct_extrating**.
- PIN_FLD_FLAGS specifies that PCM_OP_SUBSCRIPTION_PROVISION_ERA is called at purchase time.

You must include the opcode twice in a provisioning tag, once with PIN_FLD_FLAGS set to 0 and once with PIN_FLD_FLAGS set to 1, so that it runs both at purchase and cancellation time.

- PIN_FLD_STR_VAL specifies that the profile name and profile description are localized and are stored in the **/string** object under string IDs 12 and 13.
- PIN_FLD_ACCOUNT_OBJ and PIN_FLD_SERVICE_OBJ use variables. See "Variables for Parameter Values".

Variables for Parameter Values

You can use the following variables in Table 20-5 to specify certain values available only at the time the opcode is run:

Table 20-5 Run-Time Variables for Parameters

Variable	Description
$ACCOUNT$	Account POID

Table 20-5 (Cont.) Run-Time Variables for Parameters

Variable	Description
$SERVICE$	Service POID
$PRODUCT$	Product POID
$DISCOUNT$	Discount POID
$OFFERING$	POID of the purchased product or discount
$PROVTAG$	Provisioning tag

Default Provisioning Tag

The default **/config/provisioning_tag** object contains the __DEFAULT__ provisioning tag. This tag is defined in the default **pin_config_provisioning_tags.xml** file. The tag calls the following opcodes:

- PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING, on product purchase.
- PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING, on product cancellation.

If you have customized these opcodes, they set and clear fields in **/service** objects when a product is purchased and canceled. If you have not customized these opcodes, you do not need to use them. You can specify other opcodes in the provisioning tags file to perform necessary actions.

The __DEFAULT__ provisioning tag is always called for specified services when a product is purchased or canceled, so you do not need to include these opcodes in any other provisioning tags you define. See the default **pin_config_provisioning_tags.xml** file for the list of services.

Using Custom Opcodes

You can create custom opcodes to perform actions not supported by existing opcodes and specify the custom opcodes in the provisioning tags configuration file. For example, you can write an opcode to add fields to a **/service** object.

Configuring Provisioning Tags for Policy-Driven Charging

Configure the resources that BRM should track by creating provisioning tags for your services. Each provisioning tag contains the types of resources valid for that tag and information on the policy attributes which are used in provisioning policy for that resource.

Policy attributes are the configured characteristics associated with a provisioning policy. They contain the information you must provide to the specific opcode which is to process changes to the product or discount associated with that provisioning tag (for example, an offer profile threshold breach notification is required and/or the language in which the subscriber requires the notification).

> **Note:**
>
> BRM stores policy attributes as subscriber preferences (**/profile/subscriber_preferences**) objects containing the following elements:
>
> - Name (required)
> - Type (optional)
> - Value (optional)

See "Configuring Subscriber Preferences" in *BRM ECE Implementing Charging* for information on storing subscriber preferences.

About the Default Provisioning Tag Provided by BRM

BRM provides a default provisioning tag in the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file located in *BRM_Home***/sys/data/config** directory.

The first section of the definition seen in Example 20-1 shows the valid resources for the default provisioning tag. It is followed by the information on the opcodes to be called when the product or discount containing the provisioning tag is purchased or canceled and the parameters that specify the fields to be added to the opcode's input flist and the value for each field.

Collecting Data for Provisioning Tags

Collect the following data for any additional services that BRM should track for policy-driven charging.

- The name for your provisioning tag. This must be the name of the offer profile with which you associate this provisioning configuration.
- Permitted types of services
- Subscriber preferences for the service type stored in the database as **/profile/subscriber_preferences** objects.

  For more information see "Configuring Subscriber Preferences" in *BRM ECE Implementing Charging*.
- Opcodes that are to be used in association with the provisioning tag

  You must set up the following for each of the opcodes you include in a provisioning tag:

  - The name and number of each opcode to run and when the opcode must be run. These opcodes contain the business logic to perform the actual provisioning, such as creating a profile.
  - Parameters that specify the fields to be added to each opcode's input flist and the value for each field.

This information can now be configured in the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file for use with your offer profiles.

Configuring Provisioning Tags for Offer Profiles

To configure provisioning tags for use with your offer profiles:

1. Go to the *BRM_Home*/**sys/data/config** directory.
2. Open the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file an XML editor or text editor.

   > **Tip:**
   >
   > Save a copy of the default configuration file before you make any changes to it.

3. Do one of the following to update this file:
   - Edit the default provisioning tag contained in this file.
   - Add your provisioning tag by placing your definition just below the default provisioning tag (*Platinum*). Include the following for each provisioning tag:
     - Provisioning tag name

       ```
       <ProvisioningTag name="YourProvisioningTagName">
       ```

       where *YourProvisioningTagName* is the name of your offer profile.
     - Each service that is valid for the provisioning tag as a **PermittedType** element as shown in Example 20-1.
     - Specifications for the opcodes to call for this provisioning tag as shown in Example 20-1.

   > **Tip:**
   >
   > Table 20-3 lists the elements in the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file, the syntax to use for each element, and a description of how to specify each element.

4. Save the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file.

Modifying Provisioning Tags

You use provision tags as the names of offer profiles associated with products and discounts. If you modify a provision tag that is in use as an offer profile, be sure to modify the corresponding offer profile name accordingly.

See "Policy-Driven Charging" for information on offer profiles.

> **Note:**
>
> - Do not remove existing provisioning tags from **pin_config_provisioning_tags.xml** when adding new tags unless you want existing tags removed from the **/config/provisioning_tag** objects in the database.
> - When you load **pin_config_provisioning_tags.xml**, all existing instances of **/config/provisioning_tag** are removed from the database, so only the provisioning tags defined in the file when you load it will be in the database.

Loading Provisioning Tag Configurations

If you are loading provisioning tags for policy-driven charging, see "Loading Provisioning Tags for Policy-Driven Charging".

After you configure provisioning tags in the **pin_config_provisioning_tags.xml** file, load the tags into the database with the **load_config_provisioning_tags** utility.

> **Note:**
>
> The utility that loads provisioning tags into the database overwrites existing provisioning tags. When updating provisioning tags, you cannot load new provisioning tags only. You must load the complete set of provisioning tags each time you run the utility.

To load provisioning tag configurations:

1. Go to the directory in which the **pin_config_provisioning_tags.xml** file is located. The default location is *BRM_Home***/sys/data/config**.
2. Use the following command to run the **load_config_provisioning_tags** utility:

   ```
   load_config_provisioning_tags pin_config_provisioning_tags.xml
   ```

   > **Note:**
   >
   > - When you run the utility, the**pin_config_provisioning_tags.xml**and **business_configuration.xsd** files must be in the same directory. By default, both files are in *BRM_Home***/sys/data/config**.
   > - This utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

   If you do not run the utility from the directory in which **pin_config_provisioning_tags.xml** is located, include the complete path to the file, for example:

   ```
   load_config_provisioning_tags BRM_Home/sys/data/config/
   pin_config_provisioning_tags.xml
   ```

For more information, see **load_config_provisioning_tags**.

3. Stop and restart the Connection Manager (CM).
4. To verify that the provisioning tags were loaded, display the **/config/provisioning_tag** object by using the Object Browser or the **robj** command with the **testnap** utility.

Loading Provisioning Tags for Policy-Driven Charging

Use the **load_config_provisioning_tags** utility to load the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file into the BRM database.

To do so:

1. Ensure that the provisioning tags required for policy-driven charging are configured in the **pin_offer_profile_provisioning_tags_policy_attributes.xml** XML file.
2. Go to the directory in which the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file is located. The default location is *BRM_Home***/sys/data/config**.
3. Commit the provisioning tags and policy attributes XML file to the BRM database.

   ```
   load_config_provisioning_tags -d -v
   pin_offer_profile_provisioning_tags_policy_attributes.xml
   ```

   where:

   - **-d** creates a log file for debugging purposes.
   - **-v** displays information about successful or failed processing as the utility runs

   For more information on the **load_config_provisioning_tags** utility, see "load_config_provisioning_tags" in *BRM Developer's Guide*.
4. To verify that the provisioning tags were loaded, display the **/config/provisioning_tag** object by using the Object Browser or the **robj** command with the **testnap** utility.
5. Stop and restart the Connection Manager (CM).

The provisioning tags and policy attributes information associated with policy-driven charging is now stored in **/config/provisioning_tag** objects in the database.

Modifying and Compiling the Provisioning Policy Source File

If a provisioning tag defined in the provisioning tag framework uses the PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING policy opcode, you must add the tag to the **fm_subscription_pol_provisioning.c** file and recompile the file.

A provisioning tag uses PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING opcode if it is included in a product and if it uses a service associated with the __DEFAULT__ provisioning tag. See "Sample Provisioning Tag XML File" for the list of permitted services for __DEFAULT__.

Modifying and compiling the provisioning policy source file enables PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING to handle the tags.

To modify the provisioning policy source file for provisioning tags created using the provisioning tag framework:

1. Open the *BRM_Home***/source/sys/fm_subscription_pol/fm_subscription_pol_provisioning.c** file.

2. Add the provisioning tag name to the **service_info** table for the appropriate service.

   For example, to add a provisioning tag called *test* to **/service/ip**, change this code:

```
static char *tags_ip[] = {
  "example",
  NULL    /* MUST BE LAST! */
};
```

   to the following:

```
static char *tags_ip[] = {
"example",
"test",
  NULL    /* MUST BE LAST! */
};
```

3. Add code in the plp function to handle the new tag. You do this in the PROVISIONING FUNCTIONS section. The functions are grouped by service type.

   For example, to add the provisioning tag *test* to **/service/ip**, change this code:

```
static void
plp_ip(pcm_context_t *ctxp, poid_t *svc_obj_p, int32 buy,
  char *tag, pin_errbuf_t *ebufp)
{
  if (strcmp(tag, "example") == 0) {
    plp_example(ctxp, svc_obj_p, buy, tag, ebufp);
  }
  else{
    plp_ssg(ctxp, svc_obj_p, buy, tag, ebufp);
  }
}
```

   to the following:

```
static void
plp_ip(pcm_context_t *ctxp, poid_t *svc_obj_p, int32 buy,
  char *tag, pin_errbuf_t *ebufp)
{
  if (strcmp(tag, "example") == 0) {
    plp_example(ctxp, svc_obj_p, buy, tag, ebufp);
  }
  else if (strcmp(tag, "test") == 0) {
    /*skip*/
  }else{
    plp_ssg(ctxp, svc_obj_p, buy, tag, ebufp);
  }
}
```

4. Compile and save the file.

Using a Policy Source File to Set Up Provisioning

You can define provisioning tags directly in the **fm_subscription_pol_provisioning.c** file without using the provisioning tag framework. This file is the single source file for all provisioning operations.

> **Note:**
>
> The provisioning tags framework is the preferable method for creating provisioning tags.

To define provisioning tags directly in source code, follow these steps:

1. Open the *BRM_Home***/source/sys/fm_subscription_pol/fm_subscription_pol_provisioning.c** file.
2. Define your provisioning tags by following the instructions in the file.
   a. In each entry in the **provisioning_tags** table, include the name of the service associated with the tag, the tag name, and calls to service-specific functions.
   b. Ensure that each function included in the table does the following:
      - Changes fields in the service object when customers purchase the service
      - Clears the appropriate fields when customers cancel the service
3. Compile and save the file.

If you create a provisioning tag in the policy source file, you must modify and recompile the source file to modify or delete the tag.

Sample Provisioning Tag XML File

Following is the default provisioning tag XML file. This file defines the provisioning tag named __DEFAULT__, which includes several permitted services and two opcodes:

```
<?xml version="1.0" encoding="UTF-8"?>

<BusinessConfiguration
    xmlns="http://www.portal.com/schemas/BusinessConfig"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig
business_configuration.xsd">

    <ProvisioningTagConfiguration>
    <ProvisioningTagList>

        <ProvisioningTag name="__DEFAULT__">

            <PermittedTypes>/service/email</PermittedTypes>
            <PermittedTypes>/service/ip/gprs</PermittedTypes>
            <PermittedTypes>/service/content</PermittedTypes>
            <PermittedTypes>/service/vpdn</PermittedTypes>
            <PermittedTypes>/service/ip</PermittedTypes>
            <PermittedTypes>/service/ip/gprs</PermittedTypes>
            <PermittedTypes>/service/telco</PermittedTypes>
            <PermittedTypes>/service/telco/gsm</PermittedTypes>
            <PermittedTypes>/service/telco/gsm/data</PermittedTypes>
            <PermittedTypes>/service/telco/gsm/fax</PermittedTypes>
            <PermittedTypes>/service/telco/gsm/sms</PermittedTypes>
            <PermittedTypes>/service/telco/gsm/telephony</PermittedTypes>

            <OpcodeList>
                <OpcodeName>PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING</
OpcodeName>
```

```
                        <OpcodeNumber>417</OpcodeNumber>
                        <OpcodeMode>0</OpcodeMode>

                        <OpcodeParamsList>
                            <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                            <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
                        </OpcodeParamsList>
                        <OpcodeParamsList>
                            <OpcodeParamName>PIN_FLD_PROVISIONING_TAG</OpcodeParamName>
                            <OpcodeParamValue>$PROVTAG$</OpcodeParamValue>
                        </OpcodeParamsList>
                    </OpcodeList>

                    <OpcodeList>
                        <OpcodeName>PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING</OpcodeName>
                        <OpcodeNumber>418</OpcodeNumber>
                        <OpcodeMode>1</OpcodeMode>

                        <OpcodeParamsList>
                            <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                            <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
                        </OpcodeParamsList>
                        <OpcodeParamsList>
                            <OpcodeParamName>PIN_FLD_PROVISIONING_TAG</OpcodeParamName>
                            <OpcodeParamValue>$PROVTAG$</OpcodeParamValue>
                        </OpcodeParamsList>
                    </OpcodeList>

                </ProvisioningTag>

            </ProvisioningTagList>
            </ProvisioningTagConfiguration>

        </BusinessConfiguration>
```

Default Provisioning Tag for Policy-Driven Charging

Example 20-1 shows the default provisioning tag provided by BRM in the **pin_offer_profile_provisioning_tags_policy_attributes.xml** file located in the *BRM_Home*/**sys/data/config** directory.

Example 20-1 Default Provisioning Tag Configuration

```
<BusinessConfiguration xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig
business_configuration.xsd">
<ProvisioningTagConfiguration>
   <ProvisioningTagList>
     <ProvisioningTag name="Platinum">
        <PermittedTypes>/service/email</PermittedTypes>
        <PermittedTypes>/service/ip/gprs</PermittedTypes>
        <PermittedTypes>/service/content</PermittedTypes>
        <PermittedTypes>/service/vpdn</PermittedTypes>
        <PermittedTypes>/service/ip</PermittedTypes>
        <PermittedTypes>/service/fax</PermittedTypes>
        <PermittedTypes>/service/ip/gprs</PermittedTypes>
        <PermittedTypes>/service/telco</PermittedTypes>
        <PermittedTypes>/service/telco/gsm</PermittedTypes>
        <PermittedTypes>/service/telco/gsm/data</PermittedTypes>
        <PermittedTypes>/service/telco/gsm/fax</PermittedTypes>
        <PermittedTypes>/service/telco/gsm/sms</PermittedTypes>
        <PermittedTypes>/service/telco/gsm/sms</PermittedTypes>
```

```
        <OpcodeList>
            <OpcodeName>PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES</OpcodeName>
            <OpcodeNumber>3916</OpcodeNumber>
            <OpcodeMode>0</OpcodeMode>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                <OpcodeParamValue>0.0.0.0 /profile/subscriber_preferences -1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_DELETED_FLAG</OpcodeParamName>
                <OpcodeParamValue>0</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_ACCOUNT_OBJ</OpcodeParamName>
                <OpcodeParamValue>$ACCOUNT$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SERVICE_OBJ</OpcodeParamName>
                <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_NAME</OpcodeParamName>
                <OpcodeParamValue>SET_SUBSCRIBER_PREFERENCES</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SUBSCRIBER_PREFERENCES[0].PIN_FLD_NAME</OpcodeParamName>
                <OpcodeParamValue>Language</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SUBSCRIBER_PREFERENCES[0].PIN_FLD_
VALUE</OpcodeParamName>
                <OpcodeParamValue>English</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SUBSCRIBER_PREFERENCES[0].PIN_FLD_SUBSCRIBER_PREFERENCE_
ID</OpcodeParamName>
                <OpcodeParamValue>1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SUBSCRIBER_PREFERENCES[1].PIN_FLD_
NAME</OpcodeParamName>
                <OpcodeParamValue>Channel</OpcodeParamValue>
            </OpcodeParamsList>

            <OpcodeParamsList>
                 <OpcodeParamName>PIN_FLD_SUBSCRIBER_PREFERENCES[1].PIN_FLD_
VALUE</OpcodeParamName>
                 <OpcodeParamValue>IVR</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SUBSCRIBER_PREFERENCES[1].PIN_FLD_SUBSCRIBER_PREFERENCE_
ID</OpcodeParamName>
                <OpcodeParamValue>2</OpcodeParamValue>
            </OpcodeParamsList>
        </OpcodeList>
        <OpcodeList>
            <OpcodeName>PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES</OpcodeName>
            <OpcodeNumber>3916</OpcodeNumber>
            <OpcodeMode>1</OpcodeMode>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                <OpcodeParamValue>0.0.0.0 /profile/subscriber_preferences -1</OpcodeParamValue>
```

```
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_DELETED_FLAG</OpcodeParamName>
                <OpcodeParamValue>1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_ACCOUNT_OBJ</OpcodeParamName>
                <OpcodeParamValue>$ACCOUNT$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SERVICE_OBJ</OpcodeParamName>
                <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_NAME</OpcodeParamName>
                <OpcodeParamValue>DEL_SUBSCRIBER_PREFERENCES</OpcodeParamValue>
            </OpcodeParamsList>
        </OpcodeList>
    </ProvisioningTag>
  </ProvisioningTagList>
</ProvisioningTagConfiguration>
</BusinessConfiguration>
```

21

Working with Extended Rating Attributes

This chapter describes how to use extended rating attributes (ERAs) in your Oracle Communications Billing and Revenue Management (BRM) system and describes how BRM rates events using ERAs.

Topics in this document:

About Extended Rating Attributes

Extended rating attributes (ERAs) provide special rates or discounts based on a specific attribute of a service or account, such as a telephone number. For example, you use ERAs to offer special friends and family rates or a birthday discount.

You can create two types of ERAs:

- **Service-level**: An ERA that is used with a specific service, such as a telco service. Special rates for friends and family and closed user group telephone calls are examples of service-level ERAs.

  > **Note:**
  >
  > A service-level ERA can be assigned to a service that represents a subscription so that it applies to all services associated with a customer's subscription.

- **Account-level**: An ERA that can be used with any service (for example, a birthday discount).

You create an ERA by adding it to a provisioning tag.

ERAs must be configured with specific data for each customer. For example, to set up a birthday discount, you must specify the customer's birthday.

You configure ERAs in Customer Center, where ERAs are called *promotions*. ERAs you configure are stored as profiles.

For more information about ERAs, see:

- About Multiple ERA Lists
- About Configuring ERAs in Customer Center
- About Sharing ERAs

- Creating ERAs

About Multiple ERA Lists

For some types of ERAs, such as friends and family, you can create multiple lists for the same ERA. Each list is identified as a *label* in the BRM system. This enables you to set up rating and discounting to select the rate or discount based on the label name.

For example, you can design a friends and family ERA for a GSM telephony service that charges different rates for friends and family using the labels in Table 21-1:

Table 21-1 Example Labels and Rates

ERA Label	Rate
MYFAMILY	1 cent per minute
MYFRIENDS	2 cents per minute

You include labels when you define the ERA in a provisioning tag.

You use price selectors, discount selectors, or charge selectors to specify different rates or discounts for each label.

About Configuring ERAs in Customer Center

Use Customer Center to configure an ERA for an individual customer. When a customer purchases a charge offer or discount that includes a provisioning tag with an ERA, that ERA is displayed in Customer Center as a *promotion*. You specify values for the ERA on the **Promotion** tab.

When you configure an ERA in Customer Center, the values are stored in one of these profile objects:

- **/profile/serv_extrating** for a service-level ERA
- **/profile/acct_extrating** for an account-level ERA

> **Note:**
>
> – Even though an account is qualified to use ERAs, you do not have to implement them in the account.
>
> – BRM does not validate any data entered when configuring ERAs (for example, telephone numbers for the friends and family discount). To create validation rules for these entries, edit the PCM_OP_CUST_POL_PREP_PROFILE policy opcode.
>
> – ERA codes are defined in Pipeline Manager configuration files. The types of values you enter in Customer Center depends on how you configure the ERAs in Pipeline Manager. For example, if the ERA is configured to match a telephone number in an event, you would enter telephone numbers in Customer Center.
>
> – When configuring ERAs in Customer Center, use only uppercase letters, ASCII 7-bit punctuation, and no spaces.

About Sharing ERAs

You can share the ERA values you configure for one service with other services by using profile sharing groups. For example, you can share the same list of phone numbers for a friends and family ERA among all the customers whose phone numbers are included in the list.

You can share service-level ERAs only.

Creating ERAs

This section is an overview of the main steps you take to create an ERA.

Each of the following steps is a separate task. Use the links for detailed information about each task:

1. **Create a provisioning tag.** ERAs are defined in provisioning tags, which you can create in a few different ways.
2. **Define how the ERA is rated.** For example, an ERA can be rated based on the usage type or on the ERA label name.
3. **Add the ERA's provisioning tag to a charge offer or discount offer**.
4. **Add the name and description to the ERA description file.** The name and description appears in Customer Center. For provisioning tags created with the provisioning tag framework, you must take additional steps to set up the name and description. See "Customizing ERA Names and Descriptions for Client Applications ".
5. **Configure the ERA for an individual customer.** See "About Configuring ERAs in Customer Center".
6. **(Optional) Create a profile sharing group to share the ERA configuration with other accounts or services.** See "About Sharing ERAs ".

> **Note:**
>
> You can share service-level ERAs only.

Creating Friends and Family ERAs

BRM has built-in support for friends and family ERAs. This feature enables you to create an ERA with multiple lists, each containing phone numbers, APN addresses, email addresses, or other values. You can also use profile sharing to share a friends and family ERA with multiple services.

Following is a description of the steps for creating a friends and family ERA. Use the links for more detailed information about carrying out each task:

1. Create a provisioning tag.

   Create a provisioning tag with a service-level ERA called FRIENDS_FAMILY and one or more labels, which are ERA lists. The labels are optional, but they give you the flexibility to define multiple lists for an ERA and to use the label name as a rule for model selectors or charge selectors.

   You create the provisioning tag in different ways depending on its service:

- For any service, you can use the provisioning tag framework. This involves defining the tag in the **pin_config_provisioning_tags.xml** file.

  The definition of the tag includes one or more opcodes that are run when the provisioning tag is purchased. For ERAs, you can use PCM_OP_SUBSCRIPTION_PROVISION_ERA. This opcode creates a **/profile** object for the ERA.

- For telco services, use the Provisioning Tags application, or modify and load the service-specific version of the **pin_telco_tags** file.

> **Note:**
>
> For telco services, you cannot use the provisioning tag framework to include supplementary services and service extensions in the same provisioning tag with the ERA.

2. Enable **pin_notify** entries.

   a. Uncomment the following entries in the **pin_notify** file:

   ```
   3788    0      /event/group/sharing/profiles/create
   3788    0      /event/group/sharing/profiles/modify
   ```

   b. Load the **pin_notify** file and restart the Connection Manager (CM).

3. For pipeline rating, configure the ISC_ProfileAnalyzer iScript if needed.

   ISC_ProfileAnalyzer matches a value in an EDR, such as a phone number, with lists in the friends and family ERA. It then adds the names of ERA labels that include that value to the EDR.

   By default, this iScript analyzes an ERA named FRIENDS_FAMILY that uses a GSM telephony service. You can customize the iScript to handle a different service or ERA name.

   To enable ISC_ProfileAnalyzer, configure the FCT_iScript module and set the **Active** entry to **True**.

> **Note:**
>
> For rerating, ISC_ProfileAnalyzer must be enabled. The default registry file is *BRM_home***/conf/wirelessRealtime.reg**.

   See "Customizing Pipeline Rating for Friends and Family ERAs ".

4. For pipeline rating, create a price selector or discount selector.

   For pipeline events, use a price or discount selector to specify different rates or discounts for different ERA labels. See "Using ERAs with Multiple Lists".

5. Create a charge offer or discount offer that includes the provisioning tag.

   For a charge offer, include the charge with the price selector or charge selector you defined. For a discount, include the discount selector you defined.

> **Note:**
>
> - You cannot use provisioning tags created with the Services Framework Manager with discounts.
> - Use the provisioning tag framework to create an ERA that you want to add directly to a discount.

6. Add the names and descriptions to the ERA description file.

   Names and descriptions from this file are displayed in Customer Center and Provisioning Tags. See "Customizing ERA Names and Descriptions for Client Applications ".

7. Enter values for the friends and family promotion for a specific customer who purchases a charge offer or discount offer with the friends and family provisioning tag.

   > **Note:**
   >
   > If you do not add name-value pairs for friends and family ERAs, DAT_AccountBatch does not load the friends and family ERAs into the pipeline memory at startup.

   Promotion is the term used for *ERA* in Customer Center. For example, for a telephony service, you would configure the promotion by entering phone numbers. For other services, you would enter the values relevant for the service, such as APN or email addresses.

   If the ERA contains multiple lists, you configure each list in Customer Center.

8. To share ERA lists between accounts, create a profile sharing group.

9. (Optional) Create a profile sharing group to share the friends and family lists with other accounts or services.

   > **Note:**
   >
   > You can share service-level ERAs only.

   When you configure a promotion in Customer Center, the values are stored in a profile. You can share that profile with services owned by accounts that own a charge offer or discount offer with the same ERA.

   > **Note:**
   >
   > Typically, you would set up rating or discounting to use either the ERA label name with a model selector or the usage type. If you are using ERA labels and model selectors, you should comment out the IRL_UsageType iRule from the registry.

Sample Friends and Family Provisioning Tag

This is sample XML code for a provisioning tag that creates a friends and family ERA for an email service.

This code would be added to the **pin_config_provisioning_tags.xml** file:

```
<ProvisioningTag name="FRIENDS_FAMILY">

            <PermittedTypes>/service/email</PermittedTypes>

            <OpcodeList>
                <OpcodeName>PCM_OP_SUBSCRIPTION_PROVISION_ERA</OpcodeName>
                <OpcodeNumber>9066</OpcodeNumber>
                <OpcodeMode>0</OpcodeMode>

                <OpcodeParamsList>
                    <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                    <OpcodeParamValue>0.0.0.0 /profile/serv_extrating -1</
OpcodeParamValue>
                </OpcodeParamsList>

                <OpcodeParamsList>
                    <OpcodeParamName>PIN_FLD_ACCOUNT_OBJ</OpcodeParamName>
                    <OpcodeParamValue>$ACCOUNT$</OpcodeParamValue>
                </OpcodeParamsList>

                <OpcodeParamsList>
                    <OpcodeParamName>PIN_FLD_FLAGS</OpcodeParamName>
                    <OpcodeParamValue>0</OpcodeParamValue>
                </OpcodeParamsList>

                <OpcodeParamsList>
                    <OpcodeParamName>PIN_FLD_SERVICE_OBJ</OpcodeParamName>
                    <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
                </OpcodeParamsList>

                <OpcodeParamsList>
                    <OpcodeParamName>PIN_FLD_NAME</OpcodeParamName>
                    <OpcodeParamValue>FRIENDS_FAMILY</OpcodeParamValue>
                </OpcodeParamsList>

                <OpcodeParamsList>

<OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_EXTRATING.PIN_FLD_LABEL</OpcodeParamName>
                    <OpcodeParamValue>MYFRIENDS</OpcodeParamValue>
                </OpcodeParamsList>
            </OpcodeList>

            <OpcodeList>
                <OpcodeName>PCM_OP_SUBSCRIPTION_PROVISION_ERA</OpcodeName>
                <OpcodeNumber>9066</OpcodeNumber>
                <OpcodeMode>1</OpcodeMode>

                <OpcodeParamsList>
                    <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                    <OpcodeParamValue>0.0.0.0 /profile/serv_extrating -1</
OpcodeParamValue>
                </OpcodeParamsList>

                <OpcodeParamsList>
```

```
                        <OpcodeParamName>PIN_FLD_ACCOUNT_OBJ</OpcodeParamName>
                        <OpcodeParamValue>$ACCOUNT$</OpcodeParamValue>
                    </OpcodeParamsList>

                    <OpcodeParamsList>
                        <OpcodeParamName>PIN_FLD_FLAGS</OpcodeParamName>
                        <OpcodeParamValue>1</OpcodeParamValue>
                    </OpcodeParamsList>

                    <OpcodeParamsList>
                        <OpcodeParamName>PIN_FLD_SERVICE_OBJ</OpcodeParamName>
                        <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
                    </OpcodeParamsList>

                    <OpcodeParamsList>
                        <OpcodeParamName>PIN_FLD_NAME</OpcodeParamName>
                        <OpcodeParamValue>FRIENDS_FAMILY</OpcodeParamValue>
                    </OpcodeParamsList>

                    <OpcodeParamsList>

<OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_EXTRATING.PIN_FLD_LABEL</OpcodeParamName>
                        <OpcodeParamValue>MYFRIENDS</OpcodeParamValue>
                    </OpcodeParamsList>
                </OpcodeList>

</ProvisioningTag>
```

> **Note:**
>
> The PCM_OP_SUBSCRIPTION_PROVISION_ERA opcode is specified twice, once with OpcodeMode and PIN_FLD_FLAGS set to **0** to run at purchase time and once with OpcodeMode and PIN_FLD_FLAGS set to **1** to run at cancel time. This opcode requires using either **0** or **1** for these values.

To create an ERA with a second list, add two more <OpcodeList> blocks using the same opcode and the same values, except for a different value for PIN_FLD_INHERITED_INFO.PIN_FLD_EXTRATING.PIN_FLD_LABEL. Do the same if you want additional lists for the ERA.

Using ERAs with Multiple Lists

You can have multiple rates or discounts for an ERA with multiple ERA labels. An ERA label is an individual list of values, such as phone numbers or email addresses. A single ERA can have multiple ERA labels.

For example, a friends and family ERA can have separate lists for friends and family, each with a different rate. During rating or discounting, BRM chooses the correct rate based on the label name.

Using Model Selectors for ERAs

You can use price or discount selectors to define different rates or discounts for each ERA label in an ERA. Create model selector rules based on the PROFILE_LABEL_LIST EDR field. For each ERA label name, you can specify a different price or discount. During rating or

discounting, Pipeline Manager matches a value in the EDR such as a phone number with the values in the ERA labels.

Model selectors also rank the price or discounts. If the EDR matches more than one ERA label, the first pricing whose rule matches a label in the EDR is used.

In the following example, a friends and family ERA has two labels, MYFAMILY and MYFRIENDS. Each label is associated with a different pricing through a price selector, as shown in Table 21-2:

Table 21-2 Example of Friends and Family ERA

Rank	DETAIL.PROFILE_LABEL_LIST	Pricing
1	*MYFAMILY*	1_cent_per_min
2	*MYFRIENDS*	2_cent_per_min

> **Note:**
>
> To use a discount with a discount selector for a telco service, use the provisioning tag framework to create the tag with the ERA. You cannot use provisioning tags created with the Services Framework Manager with discounts.

Using Charge Selectors for ERAs

You can define different rates for each ERA label in an ERA for real-time rating by using charge selectors. To do this, use the event attribute EVENT.PIN_FLD_PROFILE_LABEL_LIST in the charge selector matrix.

In the following example, a friends and family ERA has two labels, MYFAMILY and MYFRIENDS. Each label is associated with a different rate through a charge selector, as shown in Table 21-3:

Table 21-3 Charges for ERAs

Row	EVENT.PIN_FLD_PROFILE_LABEL_LIST	Charge	Impact Category
1	*MYFAMILY*	1_cent_per_min	Default
2	*MYFRIENDS*	2_cent_per_min	Default

About Rating Based on Friends and Family ERAs

This section describes how BRM rates events with friends and family ERAs. For general information about ERAs, see "About Extended Rating Attributes". For information about working with multiple ERA lists, see "Using ERAs with Multiple Lists".

In both real-time rating and pipeline rating, the following occurs:

- If a predefined value in an event or EDR matches a value in an ERA, the names of the ERA and ERA labels are added to the event or EDR.

- If a charge selector or model selector has been defined based on the ERA label name, that selector is used to determine the rate. If an event has multiple labels, the charge selector or model selector also determines which label has priority.
- By default, friends and family ERAs are rated for telco services, but through customization they can be used and rated for any service.

For more information, see:

- Real-Time Rating for Friends and Family ERAs
- Pipeline Rating for Friends and Family ERAs

Real-Time Rating for Friends and Family ERAs

Friends and family ERAs can be rated based on either the ERA name, which is specified as the usage type, or the ERA label name.

In real-time rating, if a particular event attribute matches a value in a friends and family ERA, the ERA is added to the event's usage type field, and the ERA label is added to the event's ERA label field. An event can match more than one friends and family label.

Using the ERA label name enables you to use different rates for different values within the same ERA. For example, you can use labels to charge different rates for calls to friends and calls to families by defining a charge selector. Using the usage type, you can charge a single rate for all friends and family calls.

Following is a summary of the real-time rating process for events with friend and family ERAs. This section contains information specific to friends and family ERAs.

1. The PCM_OP_RATE_POL_PRE_RATING policy opcode calls the PCM_OP_RATE_POL_PROCESS_ERAS policy opcode, which in turn calls PCM_OP_RATE_GET_ERAS.
2. PCM_OP_RATE_GET_ERAS retrieves the valid ERAs for an event, including ERAs for the event's account, service, and shared profiles. The retrieved information includes the ERA name, label names, label values, and validity dates.
3. The PCM_OP_RATE_POL_PROCESS_ERAS policy opcode compares certain predefined event fields to the values for the ERA or ERAs valid for the event.

   By default, PCM_OP_RATE_POL_PROCESS_ERAS is configured to work with a GSM service and to compare values in the PIN_FLD_CALLING_FROM and PIN_FLD_CALLED_TO fields.

   PCM_OP_RATE_POL_PROCESS_ERAS can be easily configured to work with a GPRS service. In that case, it compares values in the PIN_FLD_ANI and PIN_FLD_DN fields.

   > **Note:**
   >
   > Using the PCM_OP_RATE_POL_PROCESS_ERAS policy opcode with services other than GSM and GPRS requires more extensive customization.

4. If a number in the PIN_FLD_CALLED_TO field matches a value in one or more of the friends and family ERA labels, PCM_OP_RATE_POL_PROCESS_ERAS adds the following to the event:
   - The ERA name to the PIN_FLD_USAGE_TYPE field. For a friends and family ERA, the name FF is added.

- The ERA label name or names to the PIN_FLD_PROFILE_LABEL_LIST field. The names are added as they were defined in the ERA (for example, MYFRIENDS and MYFAMILY). Multiple names are separated by a comma, unless you specified a different separator by customizing the opcode.

The charge or discount for the event is then calculated based on the value of either the PIN_FLD_USAGE_TYPE or PIN_FLD_PROFILE_LABEL_LIST field.

If a charge selector was used, the rating engine uses it to find the correct rate. For a friends and family ERA, typically the charge selector would use the PIN_FLD_PROFILE_LABEL_LIST event field to map to different rates. For example, you could have a different rate for each ERA label.

Prioritizing Rates for ERAs in Real-Time Rating

The charge selector also ranks the charges. If the event matches more than one ERA label, the rating engine evaluates the rates in order of their rank in the charge selector and chooses the first rate that matches a label in the EDR.

The charge selector matches ERA labels within a list of multiple labels by using the *In List* matching type. This must be selected when you create the charge selector. The rating engine then treats PIN_FLD_PROFILE_LABEL_LIST or whichever fields you include in the charge selector as a list and parses it accordingly, using a separator you specify.

In the following example, the friends and family ERA has two labels, MYFAMILY and MYFRIENDS. Each label is associated with a different rate through a charge selector, as shown in Table 21-4:

Table 21-4 Charges and Multiple Labels

Row	EVENT.PIN_FLD_PROFILE_LABEL_LIST	Charge	Impact Category
1	*MYFAMILY*	1_cent_per_min	Default
2	*MYFRIENDS*	2_cent_per_min	Default

If an event's PIN_FLD_PROFILE_LABEL_LIST is **MYFAMILY**, the rating engine chooses the **1_cent_per_min** charge. If PIN_FLD_PROFILE_LABEL_LIST contains the value **MYFAMILY, MYFRIENDS**, the rating engine again chooses the **1_cent_per_min** charge. Both charges are matched, but **1_cent_per_min** has higher priority.

Customizing Real-Time Rating for Friends and Family ERAs

You can customize the PCM_OP_RATE_POL_PROCESS_ERAS policy opcode to do the following:

- Add a service type, so you can rate ERAs for services other than telco services.

  To do this, modify the **fm_rate_pol_proc_eras_set_era_name()** function.

  To add a new service type, copy the existing code for SERVICE_TELEPHONY. To add a new service type, copy the code. In the copied code, change the service type, and change the PIN_FLD_CALLING_FROM and PIN_FLD_CALLED_TO fields to the appropriate fields for the new service type.

  This is an excerpt from the default version of **fm_rate_pol_process_eras.c** showing the code you must copy and modify:

```
if(svc && !strncmp(svc,SERVICE_TELEPHONY,strlen(SERVICE_TELEPHONY))) {
  column_flistp = PIN_FLIST_ELEM_GET(e_flistp,PIN_FLD_CALL, rec_id, 1, ebufp);
  if(column_flistp) {
    a_num = PIN_FLIST_FLD_GET( column_flistp, PIN_FLD_ANI, 1, ebufp );
    b_num = PIN_FLIST_FLD_GET( column_flistp, PIN_FLD_DNIS, 1, ebufp );
}
```

- Change the separator character used to separate multiple label names in the PIN_FLD_PROFILE_LABEL_LIST field. Comma is the default. You specify this character in the CM_OP_RATE_POL_PROCESS_ERAS policy opcode and in the charge selector. Specify the same character in both places.

Pipeline Rating for Friends and Family ERAs

In pipeline rating, if ERA labels are used and a particular EDR field matches a value in a friends and family list, the ISC_ProfileAnalyzer iScript adds the ERA label to the EDR. An EDR can match more than one friends and family label. You set up a price or discount selector to select a price or discount based on the ERA label name. See "Using Model Selectors for ERAs".

If the IRL_UsageType iRule has been configured for ERAs, and a usage type set up in Pricing Center or PCC, the ERA name (not the label name) is added to the EDR's usage type field.

Typically, you would set up rating or discounting to use either the ERA label name with a model selector or the usage type. If you are using ERA labels and model selectors, you should comment out the IRL_UsageType iRule from the registry.

Following is a summary of the pipeline rating and discounting process for events with friend and family ERAs:

1. At startup, DAT_AccountBatch gets ERA and shared profile information from the BRM database and stores it in memory.
2. When a CDR is rated, DAT_AccountBatch determines the ERAs or profile sharing group that applies to the CDR, filters out the profiles specified in the **ProfilesNotLoadedIntoEdr** registry entry, and sends the remaining profile information to the FCT_Account module.
3. FCT_Account adds ERA and shared profile data to the EDR.
4. The ISC_ProfileLabel iScript calls the search method in DAT_AccountBatch to determine if any of the profiles that were filtered contains a value that matches the EDR field value. See "Improving Pipeline Rating Performance for Events with ERAs".

   If there's a match, ISC_ProfileLabel does the following:

   a. Parses the ERA labels returned by the search method and populates them into the DETAIL.PROFILE_LABEL_LIST EDR field.

   b. Populates the DETAIL.USAGE_TYPE EDR field based on the matching ERAs found.
5. For the profiles not specified in the ProfilesNotLoadedIntoEdr registry entry, the following occurs:

   a. ISC_ProfileAnalyzer iscript analyzes the profiles and determines if any of the profiles contains a value that matches the EDR field value. If there is a match, the ERA label is added to the DETAIL.PROFILE_LABEL_LIST field in the EDR container.

      By default, ISC_ProfileAnalyzer analyzes ERA profiles named FRIENDS_FAMILY for the service codes TEL (for GSM telephony) or SMS, by comparing the value of the DETAIL.B_NUMBER (called number) EDR field to the values in the ERA. You can customize ISC_ProfileAnalyzer to analyze other ERA profiles and service types. See "Customizing Pipeline Rating for Friends and Family ERAs ".

6. The IRL_UsageType iRule adds the usage type to the DETAIL.USAGE_TYPE field in the EDR container if the conditions in the IRL_UsageType.irl file are met. For a friends and family ERA, the name added to USAGETYPE is FF.
7. For rating, FCT_MainRating finds the correct pricing for calculating the charge as follows:
   - If there is an ERA name in the USAGETYPE field, FCT_MainRating uses that ERA to rate the event.
   - If one or more ERA labels are listed in PROFILE_LABEL_LIST, FCT_MainRating chooses pricing using the rules in the price selector.
8. For discounting, FCT_DiscountAnalysis finds the correct discount to calculate the discount, using the values in the PROFILE_LABEL_LIST EDR field and the rules in the discount selector.

   The FCT_Discount module applies the discount after FCT_DiscountAnalysis determines the discount to use.

Prioritizing Rates and Discounts for ERAs in Pipeline Rating

The price or discount selector ranks the price or discounts. If the EDR matches more than one ERA label, Pipeline Manager evaluates the price or discounts in order of ranking and chooses the first price or discount whose rule matches a label in the EDR.

In the following example, the friends and family ERA has two labels, MYFAMILY and MYFRIENDS. Each label is associated with a different pricing through a price selector, as shown in Table 21-5:

Table 21-5 Price Selector

Rank	DETAIL.PROFILE_LABEL_LIST	Pricing
1	*MYFAMILY*	1_cent_per_min
2	*MYFRIENDS*	2_cent_per_min

If an EDR's DETAIL.PROFILE_LABEL_LIST is **MYFAMILY**, the rating module chooses the **1_cent_per_min** pricing. If DETAIL.PROFILE_LABEL_LIST contains the value **MYFAMILY, MYFRIENDS**, FCT_MainRating again chooses the **1_cent_per_min** pricing. Both pricings are matched, but **1_cent_per_min** has higher priority.

Improving Pipeline Rating Performance for Events with ERAs

When Pipeline Manager processes a call detail record (CDR), all the ERA profiles (owned or shared) that apply to the CDR are loaded into the event data record (EDR) container. However, some accounts may own a large number of ERA profiles or share a large number of ERA profiles in a profile sharing group. The populating of a large number of profiles into the EDR container may result in performance degradation. For example, when a Friends and Family profile sharing group contains a large number of ERA profiles that are shared with accounts in a hierarchy, all the shared profiles in the group are populated into the EDR container during CDR processing of the member accounts. In this example, the population of all the shared profiles into the EDR containers for *each* account in the hierarchy results in performance degradation.

When you have accounts that own or share a large number of ERA profiles, you can use the DAT_AccountBatch registry entry, **ProfilesNotLoadedIntoEdr**, to specify the profiles to be filtered. When a CDR is rated, if the value in the EDR field matches a value in one or more

ERA profiles in the **ProfilesNotLoadedIntoEdr** registry entry, the ISC_ProfileLabel iScript loads only the ERA labels into the EDR container.

To configure batch pipeline rating to filter ERA profiles:

1. Set the DAT_AccountBatch module registry entry **ProfilesNotLoadedIntoEdr** to the profiles that should be filtered. You can specify any number of profiles separated by a comma.

   For example:

   ```
   ProfilesNotLoadedIntoEdr {FRIENDS_FAMILY, DISCOUNTBUNDLE}
   ```

2. Configure the ISC_ProfileLabel iScript.

Customizing Pipeline Rating for Friends and Family ERAs

You customize pipeline rating for friends and family ERAs by customizing the ISC_ProfileAnalyzer and ISC_ProfileLabel iScripts.

By default, ISC_ProfileAnalyzer and ISC_ProfileLabel analyzes ERAs named FRIENDS_FAMILY for the service codes TEL (for GSM telephony) or SMS, comparing the value of the DETAIL.B_NUMBER (called number) field in the EDR to the ERA values.

To use ISC_ProfileAnalyzer and ISC_ProfileLabel with other services and ERAs, add an ELSE IF block to the onDetailEdr function. The default version of ISC_ProfileAnalyzer includes this example for adding a GPRS service:

```
else
if (edrServiceType == "GPRS")
{
   compareString = edrString(DETAIL.ASS_GPRS_EXT.APN_ADDRESS,0);
   compProfName  = "FRIENDS_FAMILY";
}
```

This block includes the entries shown in Table 21-6:

Table 21-6 Entries for GPRS Service

String	Description
edrServiceType	The service code.
compareString	The EDR field to compare to the ERA.
compProfName	The name of the ERA to analyze to compare to the EDR field.

Then you edit this information in the registry:

- Service type: Change TEL to the new service code.
- Profile name: Change FRIENDS_FAMILY to a different ERA name if needed.
- Separator: Change a comma to another character to separate multiple label names in EDRs, if needed.

Customizing ERA Names and Descriptions for Client Applications

You customize ERA names and descriptions that are displayed on the **Promotion** tab in Customer Center by editing the **era_descr.en_US** file in the *BRM_home*/**sys/msgs/eradescr** directory. You can localize this file.

> **Note:**
>
> For ERAs defined in telco provisioning tags, the names and descriptions are also used in the Provisioning Tags application.

This file includes entries for ERA names and descriptions that are identified by ID numbers. Each name and description must have a unique ID number.

You define the ID numbers differently depending on how the ERAs are defined:

- For ERAs defined in telco provisioning tags, the ID numbers must match the ID numbers used in **pin_telco_tags_gsm** or other **pin_telco_tags** files that you edit when configuring ERAs.

  For example, this is the entry for the SPECIAL_DAY account-level ERA in the **pin_telco_tags_gsm** file:

  ```
  account_era  "SPECIAL_DAY"  2  3
  ```

- For ERAs defined using the provisioning tag framework, you include each ID in the **customized.properties** file in the Customer Center SDK. You also must include the name of the ERA in the provisioning tag definition.

  See "Creating Names and Descriptions for ERAs Defined in the Provisioning Tag Framework".

The following example shows the name and description of the Special Day ERA in the **era_descr.en_US** file:

```
ID = 2 ;
VERSION = 1 ;
STRING = "Special Day" ;

ID = 3 ;
VERSION = 1 ;
STRING = "This option gives discounts on calls made on your birthday. An account can
have only one Birthday. To give this promotion, enter the word BIRTHDAY in the Name
field, and enter the customer's birthday in the Value field." ;
```

When you add or change ERA names and descriptions, you might also need to provide more detailed information for your CSRs. For example, you could provide a list of valid entries for the Customer Type ERA and instructions on when to use each entry.

After you customize the file, use the **load_localized_strings** utility to load the contents of the **era_descr.en_US** file, or a localized version of the file, into the **/strings** object.

To load the contents of **era_descr.en_US**, use this command:

```
load_localized_strings era_descr.en_US
```

> **Note:**
>
> - ERA label names are not included in the **era_descr.en_US** file and cannot be localized.
> - Default ERA names and descriptions are loaded when you install a service manager. You must load them again only if you customize them.
> - If you're loading a localized version of this file, use the correct file extension for your locale.

Creating Names and Descriptions for ERAs Defined in the Provisioning Tag Framework

If you define an ERA using the provisioning tag framework, use the following procedure to create a name and description that appears in Customer Center:

1. Add the name and description to the **era_descr.en_US** file and to any other localized versions of the ERA description file you are using.

   For information, see "Customizing ERA Names and Descriptions for Client Applications ".

   > **Note:**
   >
   > - You must use a unique ID number for each name and description you add.
   > - You must use the same ID number for the same ERA in each localized version of the ERA description file.

2. Load the names and descriptions using the **load_localized_strings** utility.

   See "Customizing ERA Names and Descriptions for Client Applications ".

3. Configure the ERA in the provisioning tag framework as follows:
   - Configure the provisioning tag to call the PCM_OP_SUBSCRIPTION_PROVISION_ERA opcode during purchase and cancel modes.
   - Specify the ERA name in the PIN_FLD_NAME parameter for the opcode. For example, for a friends and family ERA, specify **FRIENDS_FAMILY** in the provisioning tags configuration file.
   - Set the PIN_FLD_STR_VAL parameter to **12, 13** so that the profile name and profile description are localized and are stored in the **/string** object.

22

Rating Implementation and Customization

This chapter provides information about implementing and extending the default Oracle Communications Billing and Revenue Management (BRM) rating functionality.

Topics in this document:

- How Rating Works
- Managing Sessions
- Managing Price Lists
- Managing Individual Pricing Objects
- Managing Filter Sets

For general information about rating, see "About Creating a Price List".

How Rating Works

The PCM_OP_ACT_USAGE opcode is the main BRM rating opcode. In addition, subscription opcodes, such as PCM_OP_SUBSCRIPTION_CYCLE_FORWARD, are used for applying fees.

How BRM Rates and Records Usage Events

BRM uses PCM_OP_ACT_USAGE to rate usage events and record them in the BRM database. For example, this opcode is called when an event is generated for a user login session.

> **Note:**
>
> Do not call PCM_OP_ACT_USAGE directly.

PCM_OP_ACT_USAGE takes as input an **/account** object POID (Portal Object ID) and a type-only POID that specifies the type of event being recorded.

The PIN_FLD_PRODUCTS array may be included in the input flist to supply or override the list of products that the account owns.

PCM_OP_ACT_USAGE performs the following operations:

1. **Determines whether the event is pre-rated**. If the event is pre-rated, the opcode does not rate the event. However, to support taxation for the following adjustments, disputes, and settlements, this opcode performs additional filtering for pre-rated events:
   - Adjustments at the account level, subscription service level, member service level, and item level
   - Item-level disputes

- Item-level settlements

This additional filtering consists of checking the event type and performing one of the following activities, as applicable:

- **Adjustments at the account level, subscription service level, member service level:** Gets the adjustment amount and then requests deferred or immediate tax calculation for this amount.
- **Item-level adjustments, disputes, or settlements:** Evaluates the adjustment, dispute, or settlement amount for each event in the bill item and determines the proportionate amount for taxation. It then requests deferred or immediate tax calculation for this amount.

BRM chooses whether to use deferred or immediate taxation for adjustments, disputes, and settlements based on the **tax_now** entry in the Connection Manager (CM) configuration file (**pin.conf**).

For event-level disputes and settlements, PCM_OP_ACT_USAGE also processes notification events that include all the account and balance impact information required to reserve disputed resources and to release them upon settlement.

2. **Adjusts available balances to include reservations**. When PCM_OP_ACT_USAGE is called to *reauthorize* an ongoing prepaid session, an array of the resources currently reserved for the session (the PIN_FLD_RESERVATION_LIST array) is included in the input flist. PCM_OP_ACT_USAGE gets the account balances from which these reservations were made. It then temporarily adds the amount currently reserved for the session to the remaining *unreserved* amount in each retrieved balance. The sum of these two amounts represents the total amount of the resource available to the ongoing session. This amount is used to rate the reauthorization request.

   For example, customer X has a resource balance of 70 free minutes. Currently, 30 of the minutes are reserved for session A, 30 are reserved for session B, and 10 are unreserved. BRM receives a request to extend session A for 30 minutes. Before the request is rated to determine whether customer X's account has sufficient resources for it, PCM_OP_ACT_USAGE adds the free minutes currently reserved for session A (30) to the account's unreserved free minutes (10) to get the total free minutes available to session A (30 + 10 = 40). In this case, the total free minutes available to session A (40) does not cover the reauthorization request, which is for 60 minutes (30 for the initial authorization + 30 for the extension).

3. **Determines whether the event is discountable**. PCM_OP_ACT_USAGE determines whether the event is discountable.

   > **Note:**
   >
   > Discountable events include events to which you can apply product discounts, charge sharing, discount sharing, loyalty points, free minutes, and so forth.

   By default, the rating opcodes perform a credit limit check when rating an event. In CALC_ONLY mode, the credit limit check includes the effects of discounts. Because discounts can reduce the balance impact of an event that might otherwise exceed an account's credit limit, the credit limit check for discountable events is deferred to a real-time discounting pipeline.

   To determine whether an event is discountable, PCM_OP_ACT_USAGE checks the usage maps of all the discounts in your system. If at least one map contains the event type of the event being rated, the event is considered discountable.

> **Note:**
>
> PCM_OP_ACT_USAGE evaluates discount usage maps using the data that exists as of the most recent restart of the CM. Instances of event types that are newly added to discount usage maps after the most recent CM restart are therefore not found to be discountable. For example, if you add a new real-time usage event type to an existing discount that does not already include a real-time usage event, events of that type are not found to be discountable. You must restart the CM to make these newly added event types discountable.

If the event is discountable, PCM_OP_ACT_USAGE retrieves all candidate discounts associated with the customer's account. If one or more candidate discounts are returned, the PCM_OP_RATE_EVENT opcode is called with the PCM_OPFLG_RATE_WITH_DISCOUNTING flag set to **True**. This flag turns off the credit limit check performed by the real-time rating opcodes and defers it to the real-time discounting pipeline.

See "Rating Events" for more information about PCM_OP_RATE_EVENT.

4. **Determines how much to charge for the event**. If the calling opcode does not specify a rate, PCM_OP_ACT_USAGE does the following:
   - Reads the **/config/rum** object to generate a ratable usage metric (RUM) candidate for rating the event.
   - Calls PCM_OP_RATE_EVENT to rate the event. See "FM Rate Opcodes Called by PCM_OP_ACT_USAGE".
   - Calls the PCM_OP_RATE_POL_POST_RATING policy opcode to apply any modifications to the **/event/activity** object after rating the event. See "Modifying Rated Events".
5. **Applies the event's balance impact**. PCM_OP_ACT_USAGE applies the balance impact of the event to the account balances as follows:
   - If the account has resources set aside in reservations, the resources are consumed from the reservations.
   - If the account does not have reserved resources, the event balance impact is applied to a balance group in the following order based on what is provided in the input flist:
     - If a bill unit is specified in the input flist, the event balance impact is applied to the default balance group of the bill unit.
     - If a bill unit is not specified, the event balance impact is applied to the specified balance group.
     - If a balance group is not specified, the event balance impact is applied to the balance group of the specified service.
   - Otherwise, the event balance impact is applied to the default account-level balance group.

> **Note:**
>
> If the PCM_OPFLG_CALC_ONLY or PIN_ACT_CALC_MAX flag is used, the account balance is not updated.

6. **Records the event**. If no balance impact is associated with the event, PCM_OP_ACT_USAGE checks the Event Record Map configuration object (**/config/event_record_map** cached at CM startup time) against the input event type. If the input event type is configured not to be recorded in BRM, the event is not recorded. Otherwise, the event is recorded.

   PCM_OP_ACT_USAGE performs these tasks:

   - Updates the event fields using information from the specified **/account** object. PCM_OP_ACT_USAGE associates the **/item** object POID with the event and sets the PIN_FLD_CURRENCY, **/payinfo**, **/bill** object POID, and general ledger (G/L) segment ID.

     > **Note:**
     >
     > PCM_OP_ACT_USAGE does not record the PIN_FLD_EXTENDED_INFO substruct field in the event object. To record the PIN_FLD_EXTENDED_INFO substruct in the event object for customizations, modify the PCM_OP_RATE_POL_POST_RATING policy opcode.

   - Obtains information about event fields to be cached for event searches and invoicing.
   - Retrieves a list of remittance accounts that must be remitted for the event.

     > **Note:**
     >
     > BRM caches remittance specifications. The PCM_OP_REMIT_GET_PROVIDER opcode is called only when the remittance specification cache is non-null.

   - Checks whether the event is included in your system's event notification list. If it is, calls the opcodes associated with the event in the list.

Specifying the Rating Mode

You control how PCM_OP_ACT_USAGE rates and records usage events by using the following flags:

- **PCM_OPFLG_CALC_ONLY**

  When this flag is passed in the opcode call, PCM_OP_ACT_USAGE calculates the amount to charge, factoring in all applicable discounts, but does not apply the amount to the account balance.

- **PIN_ACT_CALC_MAX**

  When PIN_FLD_FLAGS is set to PIN_ACT_CALC_MAX, PCM_OP_ACT_USAGE calculates the maximum quantity that can be consumed based on the event owner's current account balance or reserved amount but does not apply the amount to the account balance.

- **PIN_RATE_FLG_RATE_ONLY**

  When PIN_FLD_FLAGS in the PIN_FLD_BAL_IMPACTS array is set to PIN_RATE_FLG_RATE_ONLY, PCM_OP_ACT_USAGE uses rated, pre-rated, and tax balance impact types to rate the event.

> **Note:**
>
> When PIN_RATE_FLG_RATE_ONLY is set, PCM_OP_ACT_USAGE returns the event's PIN_FLD_BAL_IMPACT array in the output flist.

- **PIN_RATE_FLG_RERATE**

  When the PIN_FLD_FLAGS field in the PIN_FLD_BAL_IMPACTS array is set to PIN_RATE_FLG_RERATE, PCM_OP_ACT_USAGE uses rerated tax balance impact types to rate the event.

> **Note:**
>
> When PIN_RATE_FLG_RERATE is set, PCM_OP_ACT_USAGE returns the event's PIN_FLD_BAL_IMPACT array in the output flist.

- **PIN_RATE_FLG_OVERRIDE_CREDIT_LIMIT**

  When PIN_FLD_FLAGS in the PIN_FLD_BAL_IMPACTS array is set to PIN_RATE_FLG_OVERRIDE_CREDIT_LIMIT, PCM_OP_ACT_USAGE does not check the credit limit.

FM Rate Opcodes Called by PCM_OP_ACT_USAGE

To rate events, PCM_OP_ACT_USAGE calls the following opcodes:

- PCM_OP_RATE_EVENT. See "Rating Events".
- PCM_OP_RATE_GET_PRODLIST. See "Retrieving Product Lists".

Rating Events

To rate an event, PCM_OP_RATE_EVENT does the following:

- When the optional time-stamp field PIN_FLD_WHEN_T is present in the PCM_OP_RATE_EVENT input flist, PCM_OP_RATE_EVENT searches for the price list that is valid at the time specified in the field. It uses the price list to rate the event.
- PCM_OP_RATE_EVENT sets the optional PIN_FLD_VALID_TO field for CALC_MAX and CALC_ONLY operations. This field is used to limit prepaid product authorizations to a specific time period for non-duration RUMs.
- If a product or discount starts on first usage and its validity period has not been set, selects the product or discount as a candidate for rating if the purchase time is equal to or earlier than the event time. The opcode returns the first-usage products that are used to rate the event in the PIN_FLD_FIRST_USAGE_PRODUCTS array in the output flist.
- If rating impacts a balance whose validity period has not yet been set (such as noncurrency balances that start on first usage or relative to when they are granted), calculates the resource balance validity period. If resource validity end time is restricted to the granting product's or discount's end time, this opcode compares the calculated end time with the product's or discount's end time and returns the earlier of the two in the output flist.
- Locates any rollover objects for events and returns details of the rollover object to the calling opcode.

- Determines whether it should perform a credit limit check. PCM_OP_RATE_EVENT performs a credit check unless:
  - The event is discountable or involves multiple RUMs. In CALC_ONLY mode, the credit limit check is deferred to the real-time pipeline.
  - The event being rated includes the PIN_RATE_FLG_NO_CREDIT_LIMIT_CHECK flag, the applicable rate is a credit limit override rate, or the event is a refund event. (The PIN_RATE_FLG_NO_CREDIT_LIMIT_CHECK flag is set if the event associated with the plan does not use a credit limit. You specify to not use a credit limit in Pricing Center when you set up a plan.)

  PCM_OP_RATE_EVENT sets a value in the PIN_FLD_CHECK_CREDIT_LIMIT field in its output flist to indicate whether a credit limit check should occur, where it should occur, and how the real-time discounting pipeline should handle the event if the credit limit occurs there. Flag values can be summed.
  - PIN_RATE_CREDIT_LIMIT_NEEDED (**0x1000**) indicates that a credit limit check should be performed.
  - PIN_RATE_CREDIT_LIMIT_LOCATION_CRE (**0x100**) indicates that the credit limit check is handled by PCM_OP_RATE_EVENT.
  - PIN_RATE_CREDIT_LIMIT_LOCATION_RTP (**0x200**) indicates that the credit limit check is deferred to the real-time discounting pipeline.
  - PIN_RATE_NO_CREDIT_LIMIT_DISCOUNT_ONLY (**0x0**): Indicates, for events that are processed by the real-time discounting pipeline, that no credit limit check should occur. This flag is used for events sent to the pipeline for discounting only, typically when rating in normal, as opposed to CALC_ONLY, mode.
  - PIN_RATE_CREDIT_LIMIT_CHECK (**0x1**): Indicates, for events that are processed by the real-time discounting pipeline, that a credit limit check should be performed by the FCT_CreditLimitCheck module. This flag is typically used for prepaid authorization. If the full amount cannot be authorized, FCT_CreditLimitCheck determines the maximum amount that can be authorized.
- If the event qualifies for a discount, sends the event to a real-time discounting pipeline.
- Calls the PCM_OP_RATE_TAX_CALC opcode to calculate taxes.
- When rating cycle forward events, calculates fees and refunds based on scale values, proration settings, and rates. The opcode uses the values of PIN_FLD_SCALE and PIN_FLD_ORIGINAL_SCALE passed to it by Subscription Manager to determine the scale to apply. These values are set differently depending on whether they are used for applying cycle fees or refunding them.

  The opcode calculates the cycle fee or refund based on the following formula:

  PIN_FLD_SCALE * PIN_FLD_ORIGINAL_SCALE * Rate

  The opcode retains the value of PIN_FLD_ORIGINAL_SCALE that is passed to it, but it sometimes changes the value of PIN_FLD_SCALE depending on the proration setting.

  To calculate cycle fees, the opcode sets PIN_FLD_SCALE to the following values:
  - When purchase proration is set to **Do not charge for this cycle**, PIN_FLD_SCALE is set to **0**.
  - When purchase proration is set to **Charge for the entire cycle**, PIN_FLD_SCALE is set to **1**.
  - When purchase proration is set to **Calculate the charge based on the amount used**, PIN_FLD_SCALE retains the value passed to the opcode by Subscription Manager.

  To calculate cycle fee refunds, the opcode sets PIN_FLD_SCALE to the following values:

- When cancel proration is set to **Do not charge for this cycle**, PIN_FLD_SCALE is set to **1**.
- When cancel proration is set to **Charge for the entire cycle**, PIN_FLD_SCALE is set to **0**.
- When cancel proration is set to **Calculate the charge based on the amount used**, PIN_FLD_SCALE retains the value passed to the opcode by Subscription Manager.

The opcode includes PIN_FLD_SCALE and PIN_FLD_ORIGINAL_SCALE in its output flist. These fields are stored in the **/event/billing/product/fee/cycle** object.

Retrieving Product Lists

To get the list of products for an event, PCM_OP_ACT_USAGE calls the PCM_OP_RATE_GET_PRODLIST opcode. This opcode gets a list of purchased products (**/purchased_product** objects) for an account based on the combination of service and event type in its input flist. It returns a list of base products and valid customized products.

In its initial search for products, the opcode uses the event object and optional service object in the flist to determine which products it retrieves:

- If the input flist does not include a service object POID, the opcode finds all purchased products for the account.
- If a service is included in the input flist, the opcode selects products that apply to the specified service and the event type in the flist. For example, if the event type is **/event/session/telco/gsm** and the service is **/service/telco/gsm/telephony**, the opcode finds products that apply to GSM usage.

After retrieving the list of products, the opcode checks if any of the products are overridden by a customized product that is currently valid. If a product is overridden by a valid customized product, it is removed from the list and not returned by the opcode. The valid customized product is included in the list instead.

If a product is retrieved and its validity period is not yet initialized, information about the product's purchase, cycle, and usage start and end times is returned in the purchase, cycle, and usage *_START_DETAILS and *_END_DETAILS fields respectively.

Generating Ratable Usage Metrics

Use the PCM_OP_ACT_GET_CANDIDATE_RUMS opcode to generate a ratable usage metric (RUM) that can be used to rate the event. A candidate RUM specifies the name, quantity, and unit of measurement for each resource used in the event. For example, an IP fax event's RUM might include the information in Table 22-1:

Table 22-1 Generating Ratable Usage Metrics

Name	Quantity	Unit
duration	2	minutes
page_count	10	pages
byte_count	9568	bytes

PCM_OP_ACT_GET_CANDIDATE_RUMS performs the following operations:

1. Calls the PCM_OP_ACT_POL_SPEC_CANDIDATE_RUMS policy opcode to calculate RUMs. See "Customizing How to Calculate RUMs".

2. Reads the **/config/rum** object to retrieve the list of valid RUMs for the specified event type.
3. Combines the configuration data with the event data to produce a RUM candidate for rating the event.
4. Returns the following data about each RUM:
   - RUM name
   - Quantity to rate
   - Unit (seconds, minutes, bytes)

About Calculating the Maximum Available Usage

BRM uses the PCM_OP_ACT_CALC_MAX_USAGE opcode to calculate the maximum available usage based on one of the following:

- The account's credit limit and current account balance.
- The amount reserved for the session.

For example, this opcode is called by the PCM_OP_ACT_AUTHORIZE opcode to calculate the maximum possible duration of a call to prevent prepaid customers from exceeding their current account balance.

PCM_OP_ACT_CALC_MAX_USAGE performs the following operations:

1. Specifies a huge quantity for the event RUM.
2. Calls PCM_OP_ACT_USAGE with PIN_FLD_FLAG set to PIN_ACT_CALC_MAX to rate and return the maximum quantity available based on the current balance or the reservation amount. See "How BRM Rates and Records Usage Events".
3. Returns PIN_FLD_QUANTITY set to one of the following:
   - The maximum usage allowed based on the user's available balance and credit limit or reserved amount.
   - **-1** to indicate unlimited usage. This value is returned only when *all* of the account's available resources can be consumed.

> **Note:**
>
> By default, for a duration-type RUM, PCM_OP_ACT_CALC_MAX_USAGE limits the maximum duration to 24 hours. For example, if a user's available balance enables the user to make a call longer than 24 hours, this opcode returns **-1**. To configure the duration maximum, set the **fm_act max_qty_for_duration** entry to a value greater than 24 (hours) in the CM configuration file.

This opcode propagates any errors returned from PCM_OP_ACT_USAGE.

About Currency Conversion

During rating, currency conversion is handled by the following opcodes:

- The PCM_OP_BILL_CURRENCY_CONVERT_AMOUNTS opcode converts the currency according to the conversion rate defined in the **/config/currency/conversionrates** object. You can set multiple time periods for conversion rates.

This opcode fails if the specified time is not within the time range or if the source or destination currency is invalid. The ERROR_NOT_FOUND error code is returned.

- The PCM_OP_BILL_CURRENCY_QUERY_CONVERSION_RATES opcode retrieves the conversion rates from the **/config/currency/conversionrates** object. This opcode is called by the PCM_OP_BILL_CONVERT_AMOUNTS opcode.

  This opcode fails if no conversion rate is specified between the source and destination currency types. The ERROR_NOT_FOUND error code is returned.

About Applying Cycle Forward Fees

To apply a cycle forward fee to a balance group, use the PCM_OP_SUBSCRIPTION_CYCLE_FORWARD opcode.

This opcode is called to charge or refund cycle forward fees (for example, when a product or discount is purchased, canceled, activated, or inactivated).

PCM_OP_SUBSCRIPTION_CYCLE_FORWARD performs the following operations:

1. When a product or discount is *purchased* or *activated* during the cycle, determines scale values that are used by real-time rating to prorate the cycle forward fee amount to be charged.
2. When a product or discount is *canceled* or *inactivated* during the cycle, determines scale values that are used by real-time rating to prorate the cycle forward fee amount to be refunded.
3. Uses customized products when valid to calculate fees or refunds. If a customized product is valid for only part of a cycle, it contributes toward the total charge or refund based on the length of its validity.
4. When a rate change is scheduled for the *next* or *current* cycle, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD gets a list of rates and the period in which they are applicable and calculates the correct charges.
5. Calculates and sets CYCLE_FEE_START_T and CYCLE_FEE_END_T to the next period, if applicable.
6. After all the products and discounts and their cycle forward rates are determined, it sends the information to PCM_OP_ACT_USAGE to rate and apply the charges.

> **Note:**
>
> If the product or discount starts on first usage (when the customer first uses the product or discount) and its validity period has not yet been set, PCM_OP_SUBSCRIPTION_CYCLE_FORWARD does not call PCM_OP_ACT_USAGE, and cycle fees are not applied.

7. For auditing purposes, it creates **/event/billing/product/fee/cycle/cycle_forward**_*type* objects, where *type* is the frequency of the cycle forward charge (for example, daily, weekly, monthly, bimonthly, quarterly, semiannual, or annual).
8. If successful, it returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/cycle_forward**_*type* event.

About Applying Cycle Arrears Fees

To apply a cycle arrears fee, use the PCM_OP_SUBSCRIPTION_CYCLE_ARREARS opcode.

> **Note:**
>
> Cycle arrears fees are applied only for a single month.

PCM_OP_SUBSCRIPTION_CYCLE_ARREARS performs the following operations:

1. When products or discounts with cycle fees are canceled or inactivated during a cycle, calculates the scale to prorate the cycle arrears fee amount to be charged.
2. When a rate change occurred in the *previous* or *next* cycle, gets a list of rates and the period in which they are applicable and calculates the correct charges.
3. Uses customized products when valid to calculate fees or refunds. If a customized product is valid for only part of a cycle, it contributes a portion of the total charge or refund based on the length of its validity.
4. After all the product and discount cycle arrears rates are determined, sends the information to PCM_OP_ACT_USAGE to rate and apply the charges.

   > **Note:**
   >
   > If the product or discount starts on first usage (when the customer first uses the product or discount) and its validity period has not yet been set, PCM_OP_SUBSCRIPTION_CYCLE_ARREARS does not call PCM_OP_ACT_USAGE, and cycle fees are not applied.

5. Creates the **/event/billing/product/fee/cycle/cycle_arrears** event for auditing purposes.
6. If successful, returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/cycle_arrears** event.

Customizing the Cycle Interval for Products

To customize the time interval for applying cycle forward and cycle arrears fees for a specified product, use the PCM_OP_SUBSCRIPTION_POL_SPEC_CYCLE_FEE_INTERVAL policy opcode.

This policy opcode is called by PCM_OP_SUBSCRIPTION_CYCLE_FORWARD and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS. The type of cycle event is passed in the PIN_FLD_SCOPE field in the input flist.

By default, this policy opcode is an empty hook to facilitate customization of the cycle forward and cycle arrears start (CHARGED_FROM_T) and end dates (CHARGED_TO_T) for a specific product. The start and end dates provided are used by rating opcodes to calculate the scale to determine the cycle fee amount to charge or refund.

The PIN_FLD_SCALE value in the input and output flists is the original charge scale and is used only to calculate the refund scale.

For example, if a product is purchased on April 1, 2009, with a monthly cycle forward fee of $30 and the purchase, usage, and cycle end dates are set to April 20, 2009, the cycle fee amount is based on the scale for the period April 1, 2009, to April 20, 2009 (20 days) divided by the unit interval from April 1, 2009, to April 30, 2009 (30 days). The cycle fee charged is **20/30 * $30**, or $20.

If the product is canceled on April 15, 2009, the refund amount is based on the scale for the period April 15, 2009, to April 20, 2009 (5 days) divided by the unit interval from April 1, 2009, to April 20, 2009 (20 days). Because the refund amount is refunded from the charged amount ($20), the refund is **5/20 * (20/30 * $30)**, or $5. Here, the scale value 20/30 is the original charge scale or the period the product was valid during the cycle.

To change the scale (for example if you do not want to refund the full amount), change the start and end dates. The refund scale is calculated based on the dates that you provide.

About Restricting the End Time of Granted Resources That Start on First Usage

> **Note:**
>
> This is an optional task that you perform only when you configure the validity periods of granted resources to start on first usage. For more information, see "About Balance Impacts That Become Valid on First Usage".

You can configure BRM to automatically restrict the validity period of granted resources to end no later than the end time of the product or discount that grants the resource.

When you configure resource validity to start on first usage and end on a date relative to the start date (when first usage occurs), you cannot know the actual end date when setting up the rate plan. Restricting the resource end time to the product or discount end time ensures that the resource cannot continue to be consumed after the product or discount expires.

> **Note:**
>
> When a product or discount is *canceled*, the validity period end time of resources granted by that product or discount is set to the time of the cancellation.

When you restrict resource validity end time, BRM sets the end time to the product or discount end time at the time of the grant. When the customer consumes the granted resource for the first time, the relative end time is calculated:

- If the calculated end time is *later* than the granting product or discount end time, the resource validity period uses the product or discount end time.
- If the calculated end time is *earlier* than the granting product or discount end time, the resource validity period uses the calculated end time.

> **Note:**
>
> If you restrict resource validity end time, you must do so for both real-time rating and pipeline rating.

To restrict resource validity end time, see "Configuring Real-Time Rating to Restrict Resource Validity End Time".

Configuring Real-Time Rating to Restrict Resource Validity End Time

In real-time rating, you restrict resource validity end time to the end time of the product or discount that grants the resource by modifying a field in the **multi_bal** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility.

To restrict resource validity end times to product or discount end times:

1. Run the following command, which creates an editable XML file from the **multi_bal** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

   This command creates the XML file named **bus_params_multi_bal.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for following line:

   ```
   <RestrictResourceValidityToOffer>FALSE</RestrictResourceValidityToOffer>
   ```

3. Change **FALSE** to **TRUE**.

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **multi_bal** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM multi-balance configuration.

4. Save the file and rename it from **bus_params_multi_bal.xml.out** to **bus_params_multi_bal.xml**.

5. Run the following command, which loads the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_multi_bal.xml
   ```

   Run this command from the *BRM_Home***/sys/data/config** directory (where *BRM_Home* is the directory in which BRM is installed), which includes support files used by the utility.

6. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

7. Stop and restart the CM.

8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About Applying Folds

For background information about folds, see "About Fold Events".

Use the PCM_OP_SUBSCRIPTION_CYCLE_FOLD opcode to apply balance impacts for fold events.

PCM_OP_SUBSCRIPTION_CYCLE_FOLD performs the following operations:

1. Calls the PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD policy opcode to get the order in which to fold the account's resources. By default, folds are applied in ascending order of the resource IDs.
2. Checks the value of the resource's PIN_FLD_APPLY_MODE field in the **/config/beid** object to see whether the resource is specified to be rolled over. For more information, see "Specifying Which Resources to Fold".
3. Applies folds for all of the account's balance groups and also for each resource in the balance group that is specified to be rolled over. However, if a resource ID is specified in the input flist, only that resource is folded.
4. After applying the balance impacts, creates **/event/billing/cycle/fold** event for auditing purposes.
5. If successful, returns the POIDs of the **/account** object and the **/event/billing/product/fee/cycle/fold** event.

About Applying Folds Only For Account-Level Products

If a fold is configured at account level, the fold is applicable to all the services in the account. The fold is then applied to the first service that is retrieved. If the resource balance is non-zero after applying the fold to the first service, the fold is applied again to the next service until the resource balance is zero.

To apply the fold only to account-level products, you can configure an account-level balance group to track the resource that you want to fold. The fold is then applied to only the products associated with the account-level balance group and not to all the services in the account.

Customizing How Folds Are Applied

To customize how folds are applied, use the PCM_OP_SUBSCRIPTION_POL_PRE_FOLD policy opcode.

By default, this policy opcode is an empty hook provided to facilitate any customization before folding the currency or noncurrency resources. For example, when billing is run, this policy opcode is called to verify that the **pin_cycle_fees** utility has applied cycle fees to an account.

Customizing the Order to Apply Folds

To customize the order in which folds are applied, use the PCM_OP_SUBSCRIPTION_POL_SPEC_FOLD policy opcode.

By default, resources are folded in ascending order based on the resource ID. This policy opcode enables you to change the order in which resources are folded.

For example, you can fold resources in descending order of the resource IDs. To do this, sort the PIN_FLD_BALANCES array and return the sorted array.

The following example shows the PIN_FLIST_SORT statement for sorting the balances array in descending order:

```
*out_flistp = PIN_FLIST_COPY(i_flistp, ebufp);
s_flistp = PIN_FLIST_CREATE(ebufp);
PIN_FLIST_ELEM_SET(s_flistp, (void *)NULL, PIN_FLD_BALANCES, PIN_ELEMID_ANY, ebufp);

PIN_FLIST_SORT(*out_flistp, s_flistp, 1, ebufp);
```

This policy opcode returns the contents of the input flist including the POID of the **/balance_group** object and the balance array sorted in ascending order.

Specifying Which Resources to Fold

You can specify which resources are impacted by fold events. This enables you to exclude folding resources that do not need to be folded.

To specify the resources to fold, you set the PIN_FLD_APPLY_MODE parameter for the resources in the resource configuration file (*BRM_Home*/**sys/data/pricing/example/pin_beid**).

The PIN_FLD_APPLY_MODE parameter can take the following values:

- **1** indicates the resource will be folded, if appropriate.
- **0** indicates the resource will not be folded.

By default, folds are enabled for all resources in the **pin_beid** file.

After modifying the **pin_beid** file, load the contents of the file into the **/config/beid** object in the BRM database by running the **load_pin_beid** utility.

For more information, see the **pin_beid** file and "load_pin_beid".

Customizing Which Resources to Fold When Products Are Canceled

The PCM_OP_SUBSCRIPTION_POL_PREP_FOLD policy opcode prepares the list of resources that must be folded when a product is canceled. This opcode is called when a product is canceled, and it performs the following functions:

- Checks the canceled product.
- Creates a list of resources affected by the cancellation that must be folded.
- Calls PCM_OP_SUBSCRIPTION_CYCLE_FOLD and passes the list of resources as the input.

If you do not want to fold resources after a product is canceled, customize this policy opcode by removing or commenting out the code in the **fm_subscription_pol_prep_fold.c** file.

Assigning Rate Plans and Impact Categories to Events

Use the PCM_OP_ACT_POL_SPEC_RATES policy opcode to map an event to a rate plan and impact category.

> **Note:**
>
> Ensure that your rate plan structure uses custom event analysis for the event.

The opcode that calls the PCM_OP_ACT_POL_SPEC_RATES policy opcode passes in the inherited information for an event. For example, to rate a password, the PCM_OP_CUST_SET_PASSWD opcode specifies the inherited password information. The PCM_OP_ACT_POL_SPEC_RATES policy opcode specifies that the password is to be rated.

Though most custom rating functionality is now in the rate plan selector, some special cases require you to use the PCM_OP_ACT_POL_SPEC_RATES policy opcode. In such a case, you must:

1. Map your custom events to BRM opcodes in the *BRM_Home***/sys/data/config/pin_spec_rates** file, and then load the file into the BRM database by using the **load_pin_spec_rates** utility.
2. Define your custom impact categories in the *BRM_Home***/sys/data/config/pin_impact_category** file, and then load the file into the BRM database by using the **load_pin_impact_category** utility.

This configures the PCM_OP_ACT_POL_SPEC_RATES policy opcode and the rate plan selector to return an event's rate plan and impact category to real-time rating.

Rating an Event Based on Extended Rating Attributes

An event can be rated based on extended rating attributes (ERAs) by using either the usage type or the ERA label, as follows:

- **Usage type**: You can select a specific rate plan and impact category based on the usage type of an event, such as ERAs. For example, if the usage type for an event is Friends and Family, you can rate that event with a special rate of $1.00 a minute.
- **ERA label**: You can select a specific rate plan based on the ERA label. A label is a separate list of ERA values. You can have multiple labels for a single ERA and use a rate plan selector to choose a different rate based on the label.

  For example, you can have a friends and family ERA with two labels, MYFRIENDS and MYFAMILY, and use the rate plan selector to use a different rate for each list.

For more information on ERAs, see "About Extended Rating Attributes".

To rate an event by using special rates based on an ERA, PCM_OP_ACT_USAGE calls the PCM_OP_RATE_POL_PRE_RATING policy opcode before calling PCM_OP_RATE_EVENT.

The PCM_OP_RATE_POL_PRE_RATING policy opcode calls the PCM_OP_RATE_POL_PROCESS_ERAS policy opcode, which in turn calls the PCM_OP_RATE_GET_ERAS opcode.

PCM_OP_RATE_GET_ERAS retrieves the valid ERAs for an event by performing the following operations:

1. Gets the service and account data for the event from the input flist.
2. Finds the ERA information in the associated **/profile/serv_extrating** or **/profile/acct_extrating** object. The ERA information includes the ERA name, label names, label values, and validity dates.

> **Note:**
>
> By default, BRM includes all ERAs in this process, but you can improve real-time rating performance by configuring BRM to omit checking for account-level ERAs, service-level ERAs, or both during rating.

3. Identifies profile sharing groups that the service or subscription service for the event belongs to. If membership is found and the group is active, the opcode retrieves the ERA information.
4. For shared profile groups, finds ERA information in the associated **/group/sharing/profiles** object. The ERA information includes the ERA name, label names, label values, and validity dates.
5. Checks whether the ERA is valid for the event start or end time, based on the configuration.

   > **Note:**
   >
   > The configuration for using event start time for the validity check is determined by the **use_prods_based_on_start_t** entry in the CM configuration file. See "Using Event Start Time to Determine a Product's Validity".

6. If the event time is less than the effective time of any of the profiles, searches for that profile in the audit tables to get the correct data matching the event time.

   If the ERA is found to be not valid, the opcode skips the remaining steps.
7. If the service has a subscription object, reads the service profiles of that subscription object. If the same ERA is found in the service and subscription service, the service ERA takes precedence.
8. Returns the names of valid ERAs to the PCM_OP_RATE_POL_PROCESS_ERAS policy opcode.

   The PCM_OP_RATE_POL_PROCESS_ERAS policy opcode uses the data retrieved by PCM_OP_RATE_GET_ERAS to do the following:

   - Compare certain predefined event fields to the values for the ERAs found in the service or account.

     The specific fields compared depend on the service. By default, the policy opcode is configured to work with a GSM telco service and to compare values in the PIN_FLD_CALLING_FROM and PIN_FLD_CALLED_TO fields.

     The policy opcode can be easily configured to work with a GPRS service. In that case, it compares values in the PIN_FLD_ANI and PIN_FLD_DN fields.

     > **Note:**
     >
     > Using the PCM_OP_RATE_POL_PROCESS_ERAS policy opcode with services other than GSM and GPRS requires customization. See "Customizing Real-Time Rating for Friends and Family ERAs".

   - Populate the PIN_FLD_USAGE_TYPE field with the names of the valid ERAs.

     If multiple ERAs are found for service and account profiles, the ERAs can be combined in the PIN_FLD_USAGE_TYPE field. For example, if Closed User Group (CUG) and Friends & Family (FF) are the qualified ERAs for the event, PIN_FLD_USAGE_TYPE can be set to FF_CUG. By default, PIN_FLD_USAGE_TYPE is populated based on the following qualified ERAs:

     – Friends and Family (FF)

  - Closed User Group (CUG)
  - FF + CUG (FF_CUG)

  You can customize the policy opcode to add other ERAs.

- Populate the PIN_FLD_PROFILE_LABEL_LIST field with ERA label names. Multiple names are separated by a comma, unless you specified a different separator in the opcode and the rate plan selector.
- Return the output to the PCM_OP_RATE_POL_PRE_RATING policy opcode.

9. The PCM_OP_RATE_POL_PRE_RATING policy opcode returns the output flist with the PIN_FLD_USAGE_TYPE and PIN_FLD_PROFILE_LABEL_LIST fields to the calling opcode.
10. The event is then rated or discounted based on the ERA value in either PIN_FLD_USAGE_TYPE or PIN_FLD_PROFILE_LABEL_LIST.

    If there are multiple valid ERAs and a rate plan selector is being used for the event, the rating or discounting opcode uses the rate plan selector to determine which ERA should be used for rating or discounting. The rate plan selector would use either EVENT.PIN_FLD_USAGE_TYPE or EVENT.PIN_FLD_PROFILE_LABEL_LIST to determine a rate based on the ERA.

Modifying Rated Events

Use the PCM_OP_RATE_POL_POST_RATING policy opcode to modify rated **/event** objects (for example, change the G/L ID of an event). This policy opcode is called by PCM_OP_ACT_USAGE.

The input flist matches the **/event** object that you are modifying. The output flist contains the event field to be changed in the rated object.

The following example shows how to change the G/L ID of an event:

```
STORABLE CLASS /event {
DESCR = "Objects of the event class are created to record the various "
"system-initated and user-initiated events. The events are "
"either related to a specific service or to an account. The "
"event includes generic information such as start and end "
"times as well the various charges that are incurred by the "
"account due to this event.";
SEQ_START = "1";
INT32 PIN_FLD_GL_ID {
DESCR = "GLID associated with this balance impact. "
"Don't care if 0.";
ORDER = 0;
CREATE = Optional;
MODIFY = Writable;
}
}                        CREATE = Optional;
MODIFY = Writable;
```

> **Caution:**
>
> Before calling the PCM_OP_RATE_POL_POST_RATING opcode, the PCM_OP_ACT_USAGE opcode stores the balance of a rated event in a balance group cache rather than in the BRM database. If you customize the PCM_OP_RATE_POL_POST_RATING opcode to call a standard opcode to change the balance of the rated event directly in the database or in a different cache, the balance group cache from which the PCM_OP_ACT_USAGE opcode fetches the final balance impact of the event may not reflect the change made by the PCM_OP_RATE_POL_POST_RATING opcode.

Customizing How to Calculate RUMs

Use the PCM_OP_ACT_POL_SPEC_CANDIDATE_RUMS policy opcode to define the customized policy to specify the candidate RUM.

A candidate RUM specifies the name, quantity, and unit of measurement for each resource used in the event. For example, an IP fax event's RUM might have the information in Table 22-2:

Table 22-2 Example

Name	Quantity	Unit
duration	2	minutes
page_count	10	pages
byte_count	9568	bytes

By default, the PCM_OP_ACT_POL_SPEC_CANDIDATE_RUMS policy opcode copies PIN_FLD_POID and the PIN_FLD_CANDIDATE_RUMS array to the output flist.

You can change the rules to calculate data by using **/config/rum**. By default, the PCM_OP_ACT_POL_SPEC_CANDIDATE_RUMS policy opcode supports only simple-arithmetic expressions (addition, subtraction, multiplication, and division), but you can add more complex rules to get the event quantity.

If you add a candidate RUM that has the same name as any RUM in the input flist, drop the one from the input flist to ensure that the RUM names are unique.

Rating and Recording Activity Events

BRM uses the PCM_OP_ACT_ACTIVITY opcode to record and rate activity events. This enables BRM to record any type of activity event and any details specific to the event type. Any type of user action can be recorded as an activity event, but it is especially designed to represent events that occur at a single point in time. All activity events are recorded in the system simultaneously.

> **Note:**
>
> For events related to customer service usage over a period, the PCM_OP_ACT_START_SESSION and PCM_OP_ACT_LOGIN opcodes are called. For more information on the PCM_OP_ACT_START_SESSION and PCM_OP_ACT_LOGIN opcodes, see *BRM Developer's Reference*.

PCM_OP_ACT_ACTIVITY records events for either **/account** or **/service** objects. The **/account** POID must be specified in both cases.

- When an **/account** POID is used alone, PCM_OP_ACT_ACTIVITY records the event for an **/account** object.
- When both an **/account** POID and a **/service** POID are specified, PCM_OP_ACT_ACTIVITY records the event for a **/service** object.
- When a **NULL /service** POID is specified, PCM_OP_ACT_ACTIVITY records an **/account** event.

PCM_OP_ACT_ACTIVITY records event details in the following ways:

- Using the generic **/event/activity** object, the PIN_FLD_DESCR field specifies details of the event in ASCII text format.
- If this is insufficient, an inherited object type can be created by the user from the **/event/activity** object and additional fields added to describe a specific type of event in more detail. When an event of that type occurs, the input flist to this operation must use the PIN_FLD_OBJ_TYPE field to specify the type of event object to create and to include the detailed information fields in the PIN_FLD_INHERITED_INFO substructure. This enables any amount of detail to be recorded for any number of event types.

> **Note:**
>
> The PIN_FLD_INHERITED_INFO feature requires you to supply a new storable class definition. You determine the fields necessary in the new inherited storable class type and define them by using PIN_MAKE_FLD.

BRM controls what PCM_OP_ACT_ACTIVITY returns by using the following flags:

- **PCM_OPFLG_READ_RESULT**

  When this flag is set, PCM_OP_ACT_ACTIVITY returns all fields in the event object, in addition to the POID.

- **PCM_OPFLG_CALC_ONLY**
  - When this flag is set, PCM_OP_ACT_ACTIVITY only calculates the rated amount and does not record the event in the BRM database. The opcode returns the fields that would have been used to create the event object. No fields in the database are changed, and the event object is not actually created.
  - When this flag is *not* set, PCM_OP_ACT_ACTIVITY creates the **/event/activity** object to record details of the operation.

Managing Sessions

BRM stores information about sessions in **/event/session** objects in the BRM database.

A session event differs from an activity event in that the start and end of the session event are recorded separately. A session is designed to efficiently track a customer connecting to some type of service for a period of time. For events that occur at a single point in time, the PCM_OP_ACT_ACTIVITY opcode is called. For more information on the PCM_OP_ACT_ACTIVITY opcode, see *BRM Developer's Reference*.

You use the following Activity standard opcodes to manage sessions:

- PCM_OP_ACT_LOGIN. See "Starting Sessions ".
- PCM_OP_ACT_START_SESSION. See "Recording the Start of a Session".
- PCM_OP_ACT_UPDATE_SESSION. See "Updating a Session Event".
- PCM_OP_ACT_LOGOUT. See "Recording the End of a Session ".
- PCM_OP_ACT_END_SESSION. See "Rating and Recording Session Events".
- PCM_OP_ACT_LOAD_SESSION. See "Loading Sessions in Real Time".

Starting Sessions

Use the PCM_OP_ACT_LOGIN opcode to start a customer session.

PCM_OP_ACT_LOGIN performs the following operations:

1. Calls the PCM_OP_ACT_FIND_VERIFY opcode to verify the customer's identity.
2. Calls the PCM_OP_ACT_START_SESSION opcode to start a login session. See "Recording the Start of a Session".
3. Returns an flist with the user's **/account** POID.

This opcode is usually called by PCM_CONNECT, but it is also called directly. If called by PCM_CONNECT, this opcode ignores the database number in the application's **pin.conf** file and searches all schemas for the account.

Recording the Start of a Session

Use the PCM_OP_ACT_START_SESSION opcode to record the start of a session event.

This opcode records any type of session event object including details specific to the event type. This opcode does not perform rating. Fees for the session event are calculated and applied when the session ends.

Sessions can be started for either **/account** or **/service** objects. The **/account** object must be specified for both cases.

- When the **/account** object is used alone, PCM_OP_ACT_START_SESSION starts a session for the **/account** object.
- When both **/account** and **/service** object POIDs are specified, PCM_OP_ACT_START_SESSION starts a session for the **/service** object.
- When a **NULL /service** object is specified, PCM_OP_ACT_START_SESSION starts a session for the **/account** object.

Session details can be recorded in the following ways:

- In the generic **/event/session** object, the PIN_FLD_DESCR field specifies details of the session in a text format.
- If that is insufficient, you can create an inherited object type from the **/event/session** object and add fields to describe a specific type of session in more detail.

  When a session of the specified type occurs, the input flist to PCM_OP_ACT_START_SESSION can specify the type of event object to create with the PIN_OBJ_TYPE field and include the detailed information fields in the PIN_FLD_INHERITED_INFO substructure. This enables any amount of detail to be recorded for any number of session types.

  Within the PIN_FLD_INHERITED_INFO array, the program supplies a new storable class definition. You determine the fields necessary in the new inherited storable class type and define them by using PIN_MAKE_FLD. This procedure is explained in the **pcm.h** file. PCM_OP_ACT_START_SESSION then automatically creates storable class definitions and supplies updated release files containing the new storable class definitions.

BRM controls how PCM_OP_ACT_START_SESSION records the start of a session event by using the following flags:

- **PCM_OPFLG_READ_RESULT**

  When this flag is set, PCM_OP_ACT_START_SESSION returns all fields in the event object in addition to the **/account** POID.

- **PCM_OPFLG_CALC_ONLY**
  - When this flag is set, PCM_OP_ACT_START_SESSION returns the fields that would have been used to create the event object. No fields in the database are changed, and the event object is not actually created.
  - When the flag is *not* set, PCM_OP_ACT_START_SESSION creates an **/event/session** object to record the details of the session event.

Updating a Session Event

Use the PCM_OP_ACT_UPDATE_SESSION opcode to update information in a session event.

> **Note:**
>
> The session event object must already have been created by a call to PCM_OP_ACT_START_SESSION or by an opcode that calls that opcode.

This opcode updates the PIN_FLD_DESCR field and any inherited data fields in the session object. Only the fields specified in the opcode's input flist are updated; all other fields in the **/event/session** object are left unchanged. This opcode uses the generalized PIN_FLD_INHERITED_INFO substruct so it can update **/event/session** objects of any type.

Recording the End of a Session

Use the PCM_OP_ACT_LOGOUT opcode to record the end of a login session. This opcode calls the PCM_OP_ACT_END_SESSION opcode to end the session and assess any changes.

Rating and Recording Session Events

Use the PCM_OP_ACT_END_SESSION opcode to rate and record the end of a session event.

PCM_OP_ACT_END_SESSION performs the following operations:

1. Records the session's ending time stamp.
2. Updates the PIN_FLD_DESCR field and any inherited data fields in the **/event/session** object.
3. Returns the following, depending on the flags passed in:
   - When the PCM_OPFLG_READ_RESULT flag is set, the opcode returns all fields in the event object, in addition to the POID.
   - When the PCM_OPFLG_CALC_ONLY flag is set, the opcode returns the fields that would have been used to create the event object. No fields in the database are changed, and the event object is not actually created.
   - When no flags are set, the opcode returns the POID of the event object.

Loading Sessions in Real Time

Use the PCM_OP_ACT_LOAD_SESSION opcode to create, rate, and record a session event as a single operation. This opcode is valuable as a tool to load sessions in real time.

Managing Price Lists

You specify how to charge for the services and products offered by your company by creating a price list. This data is then stored in the BRM database as pricing objects, such as **/deal**, **/plan**, and **/product** objects.

For information on how BRM uses price lists, see "About Creating a Price List".

You can create, retrieve, modify, or delete price list data individually or globally.

- To create, modify, or delete all objects in a price list, use the PCM_OP_PRICE_SET_PRICE_LIST opcode. See "Committing Price List Data to the BRM Database".
- To retrieve all objects in a price list, use the PCM_OP_PRICE_GET_PRICE_LIST opcode. See "Retrieving Price List Data from the BRM Database".
- To create, modify, or delete individual pricing objects, use the PCM_OP_PRICE_COMMIT_* standard opcodes. See "Managing Individual Pricing Objects ".

Committing Price List Data to the BRM Database

Use the PCM_OP_PRICE_SET_PRICE_LIST opcode to commit all data for a price list, including plans, products, offer profiles, and deals, to the BRM database. This opcode is called directly by Pricing Center, Customer Center, your custom client application, or the **loadpricelist** utility to load price list data into the database.

> **Note:**
>
> Do not use PCM_OP_PRICE_SET_PRICE_LIST to create or update pipeline discount data such as discount models.

PCM_OP_PRICE_SET_PRICE_LIST creates one input flist that contains information for creating, modifying, or deleting the following pricing objects:

- **/best_pricing**
- **/deal**
- **/dependency**
- **/discount**
- **/offer_profile**
- **/plan**
- **/product**
- **/sponsorship**
- **/transition**

PCM_OP_PRICE_SET_PRICE_LIST performs the following in one transaction:

- Verifies the relationships and dependencies between the pricing objects
- Calls the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate any product specification attributes defined for pricing objects. See "Defining Product Specification Attributes for Pricing Components" for more information.
- Commits all objects to the BRM database
- Prepares a list of all **/sponsorship** objects that were created or modified and generates the **/event/notification/price/sponsorships/modify** notification event
- Prepares a list of all **/product** objects that were created or modified and generates the **/event/notification/price/products/modify** notification event
- Prepares a list of all **/offer_profile** objects that were created or modified and generates the **/event/notification/price/offer_profile/modify** notification event
- Prepares a list of all **/discount** objects that were created or modified and generates the **/event/notification/price/discounts/modify** notification event

Retrieving Price List Data from the BRM Database

Use the PCM_OP_PRICE_GET_PRICE_LIST opcode to retrieve price list data from the BRM database. This opcode is called directly by the following applications to retrieve price list data:

- Pricing Center
- Customer Center
- A custom client application
- **loadpricelist**, when used with the **-r** parameter

PCM_OP_PRICE_GET_PRICE_LIST retrieves data about the following pricing objects, rebuilds the product flist, and then returns the data back to the calling application:

- **/best_pricing**
- **/deal**
- **/dependency**
- **/discount**
- **/offer_profile**
- **/plan**
- **/product**
- **/sponsorship**
- **/transition**

By default, this opcode returns all price list data. However, you can specify that the opcode retrieve only **/product**, **/discount**, or **/sponsorship** objects by passing optional input flist fields:

- To retrieve only **/product**, **/discount**, or **/sponsorship** objects, pass the PIN_FLD_FLAGS input flist field set to **0x10** for **/product** objects, **0x20** for **/discount** objects, and **0x40** for **/sponsorship** objects. You can specify multiple object types by adding these values; for example, set PIN_FLD_FLAGS to **0x30** to retrieve both **/product** and **/discount** objects. You can also set the field to **0x0008** to retrieve tailor-made products or other customized pricing data from the BRM system.
- To retrieve **/product** and **/discount** objects based on the service type, pass the PIN_FLD_PERMITTED input flist field set to the desired service type, such as **/service/email**.
- To retrieve **/product**, **/discount**, or **/sponsorship** objects based on the object modification time, pass the PIN_FLD_MOD_T input flist field set to the desired time stamp.

If the opcode retrieves customized pricing data, the output flist includes the PIN_FLD_TAILORMADE field in the product array with a value of **1**. The output flist also includes PIN_FLD_TAILORMADE_DATA, which lists the resources modified by the customization and the percentage changes.

The purchase, cycle, and usage validity periods of products and discounts are returned in the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays in the output flist. If the purchase, cycle, or usage period has a relative start or end time, the relative offset information is returned in respective START_UNIT and START_OFFSET or END_UNIT and END_OFFSET fields. The unit fields specify the type of offset unit, such as minutes, days, or accounting cycles. The offset fields specify the number of offset units in the relative period.

If the purchase, cycle, or usage period starts on first usage (when the customer first uses the product or discount) and the validity period has not yet been set, the START_OFFSET field has a value of **-1**.

The validity period of resources is returned in the PIN_FLD_BAL_IMPACTS array in the PIN_FLD_RATES array. If the resource has a relative start or end time, the relative offset information is returned in the PIN_FLD_RELATIVE_START_UNIT and PIN_FLD_RELATIVE_START_OFFSET or PIN_FLD_RELATIVE_END_UNIT and PIN_FLD_RELATIVE_END_OFFSET fields. For resources, the unit fields specify only cycles. The offset fields specify the number of offset cycles in the relative period.

If the resource starts on first usage (when the customer consumes the resource for the first time) and the validity period has not yet been set, the PIN_FLD_RELATIVE_START_OFFSET field has a value of **-1**.

Managing Individual Pricing Objects

You can manage an individual pricing object, such as a **/deal** object, by using the PCM_OP_PRICE_COMMIT_* standard opcodes. These opcodes create, modify, or delete objects by using data in the input flist only; they do not verify relationships or dependencies with other pricing objects before committing changes to the database. For example, the PCM_OP_PRICE_COMMIT_PRODUCT opcode creates a **/product** object without first verifying that it is associated with a deal in the price list.

> **Note:**
>
> Because these opcodes do not verify dependencies with other pricing objects, you should use them in development environments only. To create pricing objects in a production environment, use the PCM_OP_PRICE_SET_PRICE_LIST opcode. See "Committing Price List Data to the BRM Database".

You use the following opcodes to manage individual pricing objects:

- PCM_OP_PRICE_COMMIT_PRODUCT. See "Managing /product Objects".
- PCM_OP_PRICE_COMMIT_DISCOUNT. See "Managing /discount Objects".
- PCM_OP_PRICE_COMMIT_DEAL. See "Managing /deal Objects".
- PCM_OP_PRICE_COMMIT_PLAN. See "Managing /plan Objects".
- PCM_OP_PRICE_COMMIT_DEPENDENCY. See "Managing /dependency Objects".
- PCM_OP_PRICE_COMMIT_TRANSITION. See "Managing /transition Objects".
- PCM_OP_PRICE_COMMIT_PLAN_LIST. See "Managing /group/plan_list Objects".
- PCM_OP_PRICE_COMMIT_SPONSORSHIP. See "Managing /sponsorship Objects".

> **Note:**
>
> Each of these standard opcodes calls PREP and VALID policy opcodes before committing changes to the database. You can use these policy opcodes to add data to the objects or to perform custom validation.

Managing /product Objects

Use the PCM_OP_PRICE_COMMIT_PRODUCT opcode to create, modify, or delete **/product** objects in the BRM database.

To create or modify **/product** objects, pass details about the product in the opcode's PIN_FLD_PRODUCT input flist array. In the array, you must also pass in the PIN_FLD_CODE field set to the **/product** object's unique identifier and the PIN_FLD_NAME field set to the product's name, which is modifiable.

This opcode validates and commits the following objects from a product flist: **/rate**, **/rate_plan**, **/rate_plan_selector**, and **/rollover**. Products are created or modified and, if the delete flag is sent in, deleted.

/rate_plan objects can have multiple rates, one for each rate tier (priority), date, day, and time-of-day range. These ranges are structured into a four-level tree of rate tier, date, day, and time. An optional rate plan selector is used when multiple rate plans from which to select are available.

When a product grants a noncurrency resource, such as free minutes, the validity period of the granted resource is stored in the **/rate** object's PIN_FLD_BAL_IMPACTS array. See "Managing the Validity Period of Granted Resources" for more information.

PCM_OP_PRICE_COMMIT_PRODUCT performs the following in one transaction:

- Generates the **/event/notification/price/products/modify** notification event
- Calls the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate any product specification attributes defined for **/product** objects. See "Defining Product Specification Attributes for Pricing Components" for more information.
- Commits all objects to the BRM database
- Returns the following in the output flist:
  - If all operations are successful, this opcode returns PIN_FLD_RESULTS set to **1** and PIN_FLD_PRODUCTS set to the **/product** POID.
  - If any operation fails, this opcode returns PIN_FLD_RESULTS set to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Managing the Validity Period of Granted Resources

Resource validity periods define when a granted resource is available for consumption.

To set a resource's start and end times, pass the following values in the PIN_FLD_BAL_IMPACTS array in the PIN_FLD_RATES array of the PCM_OP_PRICE_COMMIT_PRODUCT input flist:

- Set the PIN_FLD_FLAGS field to PIN_RATE_BAL_FLG_GRANTABLE (**0x08**). This specifies that the balance impact is granting the resource.
- Specify the start times as follows:
  - To start on a specific date, pass the date in PIN_FLD_START_T.
  - To start immediately, set the value of PIN_FLD_START_T to **0**.
  - To start on first usage (when the customer impacts the resource balance for the first time), set the value of PIN_FLD_RELATIVE_START_OFFSET to **-1**.
  - To start relative to the grant date, set the value of PIN_FLD_RELATIVE_START_UNIT to the type of relative unit and set the number of units in PIN_FLD_RELATIVE_START_OFFSET.
- Specify the end time as follows:
  - To never end, set the value of PIN_FLD_END_T to **0**.
  - To end relative to the start date, set the value of PIN_FLD_RELATIVE_END_UNIT to the type of relative unit and set the number of units in PIN_FLD_RELATIVE_END_OFFSET.

For more information about how to specify the unit and offset values, see "Specifying Relative Start and End Times for Granted Resources".

When the deal is committed to the BRM database, any relative validity information is stored in the PIN_FLD_START_DETAILS and PIN_FLD_END_DETAILS fields in the **/rate** object's PIN_FLD_BAL_IMPACTS array.

When the resource is granted to an account, any relative validity information is stored in the PIN_FLD_VALID_FROM_DETAILS and PIN_FLD_VALID_TO_DETAILS fields in the account **/balance_group** object's PIN_FLD_BAL_IMPACTS array.

For information about the values stored in details fields, see "Storing Start and End Times for Granted Resources".

Specifying Relative Start and End Times for Granted Resources

Relative start and end date information for granted resources is passed in UNIT and OFFSET fields:

- PIN_FLD_RELATIVE_START_UNIT
- PIN_FLD_RELATIVE_START_OFFSET
- PIN_FLD_RELATIVE_END_UNIT
- PIN_FLD_RELATIVE_END_OFFSET

The UNIT fields specify the type of offset unit, and the offset fields specify the number of units in the relative period. The UNIT fields can be one of the following values:

- 1 = Seconds
- 2 = Minutes
- 3 = Hours
- 4 = Days
- 5 = Months
- 8 = Accounting cycles
- 9 = Event cycles

The UNIT and OFFSET fields are used in combination to determine the relative offset period. For example:

- If PIN_FLD_RELATIVE_START_UNIT has a value of **8** (accounting cycles) and PIN_FLD_RELATIVE_START_OFFSET has a value of **2**, the granted resource becomes valid (available for consumption) two accounting cycles after the resource is granted.
- If PIN_FLD_RELATIVE_END_UNIT has a value of **5** (months) and PIN_FLD_RELATIVE_END_OFFSET has a value of **2**, the validity period ends two months from the date it becomes valid.

You can specify any number of units in the OFFSET fields, up to 1048576. This is equivalent to approximately 12 days in seconds, 728 days in minutes, and 119 years in hours.

Storing Start and End Times for Granted Resources

The validity period information for granted resources that is passed in the unit and offset fields of opcode flists is stored in the **/rate** object's PIN_FLD_BAL_IMPACTS array in these details fields:

- PIN_FLD_START_DETAILS
- PIN_FLD_END_DETAILS

The details fields are 32-bit integer fields that store the following values:

- **Mode**: The mode specifies generally when the validity period starts or ends. The mode also helps to determine how the start and end times are set in the account's **/balance_group** object when the resource is granted. The mode is stored in the lower 8th bit and can have one of the following values:

  For START-DETAILS:

  - **0** = PIN_VALIDITY_ABSOLUTE: The validity period starts on the date specified in PIN_FLD_START_T. When the resource is granted, the start date is set in the **/balance_group** object's PIN_FLD_VALID_FROM field.
  - **1** = PIN_VALIDITY_IMMEDIATE: The validity period starts when the resource is granted. The **/balance_group** object's PIN_FLD_VALID_FROM field is set to the grant date.
  - **3** = PIN_VALIDITY_FIRST_USAGE: The validity period starts when the resource balance is first impacted by an event.
  - **4** = PIN_VALIDITY_RELATIVE: The validity period starts relative to the grant date. When the resource is granted, the start date is calculated by adding the relative offset period to the grant date. The start date is then set in the **/balance_group** object's PIN_FLD_VALID_FROM field.

  For END-DETAILS:

  - **0** = PIN_VALIDITY_ABSOLUTE: The validity period ends on the date specified in PIN_FLD_END_T. When the resource is granted, the end date is set in the **/balance_group** object's PIN_FLD_VALID_TO field.
  - **2** = PIN_VALIDITY_NEVER: The validity period never ends. When the resource is granted, the **/balance_group** object's PIN_FLD_*_END_T field is set to **0**.
  - **4** = PIN_VALIDITY_RELATIVE: The validity period ends relative to when it starts. When the resource is granted, the end date is calculated by adding the relative offset period to the start date. The end date is then set in the **/balance_group** object's PIN_FLD_VALID_TO field.

  > **Note:**
  >
  > When the resource starts on first usage, the mode of the end time can be only PIN_VALIDITY_NEVER or PIN_VALIDITY_RELATIVE.
  >
  > When the resource is granted and its validity is activated, PIN_FLD_VALID_FROM and PIN_FLD_VALID_TO are set in the **/balance_group** object, and the mode in the PIN_FLD_VALID_FROM_DETAILS and PIN_FLD_VALID_TO_DETAILS fields in the **/balance_group** object are set to PIN_VALIDITY_ABSOLUTE.

- **Relative offset unit**: This value is set when the resource balance starts at a time relative to when it is granted or when the resource balance ends at a time relative to the start time. It specifies the type of offset unit and corresponds to the value of PIN_FLD_RELATIVE_START_UNIT or PIN_FLD_RELATIVE_END_UNIT that is passed in opcode flists. The relative offset unit is stored in the next four bits of the details field.
- **Number of offset units**: This value specifies how many offset units are in the relative period and corresponds to the value of PIN_FLD_RELATIVE_START_OFFSET or PIN_FLD_RELATIVE_END_OFFSET that is passed in opcode flists. The number of offset units is stored in the remaining 20 bits of the details field.

Modifying Rate Types

Use the PCM_OP_PRICE_SET_PRICE_LIST opcode to modify the type of a rate in a **/product** object. The rate type controls whether customers can exceed their credit limit.

To modify the **PIN_FLD_TYPE** field in **/rate** objects, pass details about the product and rate in the opcode's PIN_FLD_PRODUCT input flist array.

You can set the type to any of the following:

- **0**: The credit limit is enforced. The subscription succeeds and the resources are prorated according to the available balance.
- **740**: The credit limit is exceeded. The subscription succeeds and all available balance is used. The remaining amount is recorded as an outstanding amount to be paid at the next top up for prepaid customers or current bill for postpaid customers.
- **742**: The credit limit is exceeded. The subscription succeeds. If the customer is eligible for a loan, all available balance is used and a loan is granted for the remaining amount. If the customer is not eligible for a loan, the subscription fails.
- **743**: The credit limit is exceeded. The subscription succeeds and all available balance is used. The remaining amount is recorded as an outstanding amount to be paid at the next top up for prepaid customers or current bill for postpaid customers. This option is intended for prepaid customers.
- **744**: The credit limit is exceeded. The subscription succeeds without using the available balance. The entire amount is recorded as an outstanding amount to be paid at the next top up for prepaid customers or current bill for postpaid customers.
- **745**: The credit limit is enforced. The subscription fails and after a configured maximum number of retries, a notification event is sent to an external system for further processing.
- **746**: The credit limit is exceeded. The subscription succeeds without using the available balance. Billing will be skipped for this cycle; during rating all balance impacts are dropped and the unrated quantity is set to 0. The cycle forward date is moved to next cycle.

For example, to set the rate type to 746 for a product, you would use the following input flist:

```
0 PIN_FLD_PRODUCTS      ARRAY [0] allocated 20, used 20
1     PIN_FLD_RATE_PLANS    ARRAY [0] allocated 20, used 9
2         PIN_FLD_TAX_WHEN       ENUM [0] 0
2         PIN_FLD_CYCLE_FEE_FLAGS    INT [0] 0
2         PIN_FLD_CURRENCY        INT [0] 840
2         PIN_FLD_NAME            STR [0] "cycle_forward_monthly"
2         PIN_FLD_RATES         ARRAY [0] allocated 20, used 7
3             PIN_FLD_STEP_RESOURCE_ID    INT [0] 840
3             PIN_FLD_PRORATE_LAST   ENUM [0] 701
3             PIN_FLD_PRORATE_FIRST   ENUM [0] 702
3             PIN_FLD_TYPE           ENUM [0] 746
```

Customizing How to Create and Delete Products

To customize how to create and delete products:

- To enable data modification during **/product** object creation, use the PCM_OP_PRICE_POL_PREP_PRODUCT policy opcode. Use this policy opcode to enhance **/product** objects with additional data not provided by either the GUI application or by the Price List FM.

The PCM_OP_PRICE_COMMIT_PRODUCT opcode calls this policy opcode before creating a new **/product** object.

> **Note:**
>
> This policy opcode is called before the Price List FM performs any validation checks.

- To enable validation during **/product** object creation, use the PCM_OP_PRICE_POL_VALID_PRODUCT policy opcode. This policy opcode can be used to perform validations in addition to those performed by the Price List FM.

  The PCM_OP_PRICE_COMMIT_PRODUCT opcode calls this policy opcode before creating a **/product** object. By default, this policy opcode is an empty hook that returns the result PIN_PRICE_VERIFY_PASSED.

- To verify that deleting a **/product** object is permitted, use the PCM_OP_PRICE_POL_DELETE_PRODUCT policy opcode. Use this policy opcode to perform validations in addition to those performed by the Price List FM.

  PCM_OP_PRICE_COMMIT_PRODUCT calls this policy opcode before deleting the **/product** object. By default, it returns the result PIN_PRICE_VERIFY_PASSED.

  Success or failure is indicated by the value returned in the PIN_FLD_RESULTS output flist field: a value of **1** indicates that the operation succeeded and a value of **0** indicates that the operation failed.

Using Opcodes to Customize Products

You can customize products by changing individual rates and price models for specific resources. Customer service representatives (CSRs) can use this capability in Customer Center, but you can also call opcodes from third-party CRM applications.

> **Note:**
>
> Customer Center enforces permissioning requirements for product customization. Permissioning is not enforced when you customize products by calling opcodes directly. You must implement permissioning in the CRM tool that calls the opcodes.

Creating a Customized Product

To create custom products:

- Call the PCM_OP_PRICE_GET_PRODUCT_INFO opcode to retrieve information about the base product, based on its POID.
- Based on information about the base product and on the customization information (resources modified and the percentage modification), use the PCM_OP_PRICE_PREP_TAILORMADE_PRODUCT opcode to prepare an flist for the new customized **/product** object.
- Call the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode to purchase the customized product and create a **/purchased_product** object to represent it in the account. This opcode calls the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT opcode.

- Pass the flist prepared by PCM_OP_PRICE_PREP_TAILORMADE_PRODUCT to the PCM_OP_PRICE_SET_PRICE_LIST opcode to commit the customized **/product** object to the BRM database. See "Committing Price List Data to the BRM Database".
- Call the PCM_OP_SUBSCRIPTION_CYCLE_FORWARD opcode to apply the appropriate cycle fees. See "About Applying Cycle Forward Fees".

Modifying a Customized Product

From Customer Center, the only change you can make to an existing customized product is to modify its validity dates. This limitation does not apply when modifying customized products by calling opcodes directly, however.

- Specify changes to the customized product's rates and price models and pass them to PCM_OP_PRICE_PREP_TAILORMADE_PRODUCT to prepare an flist for the modified customized **/product** object.
- Pass the flist to PCM_OP_PRICE_SET_PRICE_LIST to update the **/product** object. See "Committing Price List Data to the BRM Database".
- Call the PCM_OP_SUBSCRIPTION_SET_PRODINFO opcode to update the **/purchased_product** object associated with the customized product.

Setting Full Day Proration

Use the PCM_OP_PRICE_SET_PRICE_LIST opcode to configure full day proration in the **/product** object by using the PIN_FLD_OFFER_VALIDITY_ROUNDING and PIN_FLD_SCALE_ROUNDING fields.

PIN_FLD_OFFER_VALIDITY_ROUNDING specifies whether to start the charge offer's validity period at the purchase time or midnight of the purchase day.

Set the PIN_FLD_OFFER_VALIDITY_ROUNDING field to one of the following:

- **0**: Uses your company's systemwide setting.
- **1**: Starts at midnight (00:00:00) of the day that the charge offer is purchased.
- **2**: Starts at the time of purchase.

PIN_FLD_SCALE_ROUNDING specifies how to calculate the scale.

Set the PIN_FLD_SCALE_ROUNDING field to one of the following:

- **0**: Calculate it based on the PIN_FLD_SCALE_ROUNDING setting.
- **1**: Calculate it based on full days.

Managing /discount Objects

Use the PCM_OP_PRICE_COMMIT_DISCOUNT opcode to:

- Create, modify, or delete **/discount** objects in the BRM database
- Create pipeline discount models

> **Note:**
>
> The opcode cannot modify pipeline discount models.

Creating /discount Objects

To create **/discount** objects, pass details about the discount in the opcode's PIN_FLD_DISCOUNT input flist array. You can also optionally create a pipeline discount model by using the PIN_FLD_PIPELINE_DISC_MODELS input flist array. See "Creating Pipeline Discount Models" for more information.

An offer profile and the provisioning tag for the associated discount use the same name and that name must be unique. If you create an offer profile to associate with existing discount, use the provisioning tag in the discount object to name the offer profile. When you configure a new discount around an existing offer profiles, use the appropriate offer profile name as the provisioning tag for the discount. Use the resource specified in the offer profile as a tracking resource in the discount that is attached to the offer profile.

When the **/discount** object does not exist in the database, PCM_OP_PRICE_COMMIT_DISCOUNT creates a **/discount** object by doing the following:

- Calling the PCM_OP_PRICE_POL_PREP_DISCOUNT policy opcode to perform any custom preparation. By default, this policy opcode is an empty hook, but you can customize it to enhance **/discount** objects with additional data or to perform other custom actions. See "Customizing /discount Objects" for more information.
- Calling the PCM_OP_PRICE_POL_VALID_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions. See "Customizing /discount Objects" for more information.
- Calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate any product specification attributes defined for **/discount** objects. See "Defining Product Specification Attributes for Pricing Components" for more information.
- Creating the specified **/discount** object in the BRM database.
- Generating the **/event/notification/price/discounts/modify** notification event. This enables you to synchronize discounts between BRM and external CRM applications.

PCM_OP_PRICE_COMMIT_DISCOUNT performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_DISCOUNT sets PIN_FLD_RESULTS to **1**.
- When any operation fails, PCM_OP_PRICE_COMMIT_DISCOUNT sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Creating Pipeline Discount Models

To create pipeline discount models, pass details about the discount model in the PCM_OP_PRICE_COMMIT_DISCOUNT opcode's PIN_FLD_PIPELINE_DISC_MODELS input flist array. The array can contain the following information:

- Discount model version and configuration
- Discount/chargeshare trigger
- Discount/chargeshare condition
- Discount/chargeshare rules

- Discount/chargeshare master
- Discount/chargeshare detail
- Discount/chargeshare step

The discount model, trigger, rule, step, and master are uniquely identified by a code string in the PIN_FLD_CODE_STR field. If a code string is not passed in the input flist, the opcode generates one automatically.

Modifying /discount Objects

To modify **/discount** objects, pass details about the discount in the opcode's PIN_FLD_DISCOUNT input flist array. In the array, you must also pass in the PIN_FLD_CODE field set to the **/discount** object's unique identifier and the PIN_FLD_NAME field set to the discount's name, which is modifiable.

> **Note:**
>
> This opcode overwrites data in existing **/discount** objects, so be sure you pass in the correct object to modify.

When the object already exists in the database, PCM_OP_PRICE_COMMIT_DISCOUNT modifies the **/discount** object by doing the following:

- Overwriting the specified **/discount** object in the BRM database.
- Generating the **/event/notification/price/discounts/modify** notification event. This enables you to synchronize discounts between BRM and external CRM applications.

PCM_OP_PRICE_COMMIT_DISCOUNT performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_DISCOUNT sets PIN_FLD_RESULTS to **1**.
- When any operation fails, PCM_OP_PRICE_COMMIT_DISCOUNT sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Deleting /discount Objects

To delete **/discount** objects, pass details about the discount in the opcode's PIN_FLD_DISCOUNT input flist array and set the PIN_FLD_DELETED_FLAG input flist field to **1**.

When PIN_FLD_DELETED_FLAG is set, PCM_OP_PRICE_COMMIT_DISCOUNT deletes the **/discount** object by doing the following:

- Calling the PCM_OP_PRICE_POL_DELETE_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
- Deleting the specified **/discount** object in the BRM database.

PCM_OP_PRICE_COMMIT_DISCOUNT performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_DISCOUNT sets PIN_FLD_RESULTS to **1**.

- When any operation fails, PCM_OP_PRICE_COMMIT_DISCOUNT sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Customizing /discount Objects

You can customize the **/discount** object before it is committed to the database by using the Price List FM policy opcodes:

- To enable data modification during **/discount** object creation, use the PCM_OP_PRICE_POL_PREP_DISCOUNT policy opcode. Use this policy opcode to enhance **/discount** objects with additional data not provided by either the GUI application or by the Price List FM.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_DISCOUNT. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  Success or failure is indicated by the value returned in the PIN_FLD_RESULTS output flist field: a value of **1** indicates that all operations succeeded and a value of **0** indicates that the operation failed.

- To enable validation during **/discount** object creation, use the PCM_OP_PRICE_POL_VALID_DISCOUNT policy opcode. This policy opcode can be used to enhance **/discount** objects with additional data not provided by either Pricing Center or by other opcodes in the Price List FM.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_DISCOUNT before creating a **/discount** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  This policy opcode is called before the Price List FM performs any default validation checks.

  Success or failure is indicated by the value returned in the PIN_FLD_RESULTS output flist field: a value of **1** indicates that all operations succeeded and a value of **0** indicates that the operation failed.

- To verify that deleting a **/discount** object is permitted, use the PCM_OP_PRICE_POL_DELETE_DISCOUNT policy opcode. Use this policy opcode to perform validations in addition to those performed by the Price List FM.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_DISCOUNT before it deletes a **/discount** object. By default, this policy opcode is an empty hook, but you can customize it to enhance **/discount** objects with additional data or to perform other customizations.

  This policy opcode is intended as a place for you to add validation checks or other functionality before allowing a **/discount** object to be deleted. For example, you may want to confirm that the valid time period for the **/discount** object has expired before allowing it to be deleted.

  Success or failure is indicated by the value returned in the PIN_FLD_RESULTS output flist field: a value of **1** indicates that all operations succeeded and a value of **0** indicates that the operation failed.

Retrieving Discount Data

Use the PCM_OP_PRICE_GET_DISCOUNT_INFO opcode to retrieve details about the real-time **/discount** object along with the pipeline discount model information from the BRM database. The discount model data includes the following:

- Discount model version and configuration
- Discount/chargeshare trigger
- Discount/chargeshare condition
- Discount/chargeshare rules
- Discount/chargeshare master
- Discount/chargeshare detail
- Discount/chargeshare step
- Balance impact

This opcode takes as input the discount object POID and optionally the discount object name. This opcode performs the following:

1. Retrieves the real-time **/discount** object details from the BRM database.
2. Retrieves the related pipeline discount model information used by the **/discount** object.
3. Returns details about the real-time **/discount** object and pipeline discount model in the output flist.

Managing /deal Objects

Use the PCM_OP_PRICE_COMMIT_DEAL opcode to create, change, or delete **/deal** objects in the BRM database.

This opcode accepts an flist consisting of a PIN_FLD_DEAL array, each element of which represents a **/deal** object.

PCM_OP_PRICE_COMMIT_DEAL determines whether to create, modify, or delete the specified **/deal** object:

- When the object does not exist, this opcode *creates* the specified **/deal** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_PREP_DEAL policy opcode to perform any custom preparation. See "Customizing How to Create and Delete Deals" for more information.
  - Calling the PCM_OP_PRICE_POL_VALID_DEAL policy opcode to perform any custom validation. See "Customizing How to Create and Delete Deals" for more information.
  - Calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate any product specification attributes defined for **/deal** objects. See "Defining Product Specification Attributes for Pricing Components" for more information.
  - Creating the specified object in the BRM database.
- When the object already exists, this opcode *modifies* the specified **/deal** object in the BRM database.

> **Note:**
>
> PCM_OP_PRICE_COMMIT_DEAL overwrites data in existing **/deal** objects, so be sure you pass in the correct object to modify.

- When the PIN_FLD_DELETED_FLAG is set, this opcode *deletes* the specified **/deal** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_DELETE_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Deleting the specified object in the BRM database.

PCM_OP_PRICE_COMMIT_DEAL performs all operations within a single transaction, so success or failure is indicated by the value returned in the PIN_FLD_RESULTS output flist field: a value of **1** indicates that all operations succeeded and a value of **0** indicates that the operation failed. When the operation fails, the opcode also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

To set the purchase, cycle, and usage start and end dates of the products and discounts in the deal, pass the following values in opcode input flists in the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays. In the following fields, the asterisk (*) indicates either PURCHASE, CYCLE, or USAGE.

> **Note:**
>
> The cycle and usage validity periods must fall within the purchase validity period.

- To start immediately, set the value of PIN_FLD_*_START_OFFSET to **0**.
- To start on first usage (when the customer uses the product or discount for the first time), set the value of PIN_FLD_*_START_OFFSET to **-1**.
- To start relative to the purchase date, set the relative unit (for example, days or cycles) in PIN_FLD_*_START_UNIT and set the number of relative units in PIN_FLD_*_START_OFFSET.

> **Note:**
>
> If you specify a relative unit for more than one period (purchase, cycle, or usage) and you select a different relative unit for each, if one of the relative units is cycles, the PCM_OP_PRICE_POL_VALID_DEAL policy opcode cannot validate that the cycle or usage period falls within the purchase period. The validation for this kind of configuration is performed when the product is purchased by a customer.

- To never end, set the value of PIN_FLD_*_END_OFFSET to **0**.
- To end relative to the start date, set the relative unit (for example, days or cycles) in PIN_FLD_*_END_UNIT and set the number of relative units in PIN_FLD_*_END_OFFSET.

For more information about how to specify the unit and offset values, see "Specifying Relative Start and End Times for Products and Discounts in a Deal" for more information.

If the deal's products or discounts grant resources that start on first usage (when the customer impacts the resource balance for the first time), you can specify that the validity period of all resources in the deal that start on first usage are set when one of those resources is impacted for the first time. To do this, set the PIN_FLD_GRANT_RESOURCES_AS_GROUP flag value in the PIN_FLD_FLAGS field.

When the deal is committed to the BRM database, the purchase, cycle, and usage validity information is stored in the PIN_FLD_*_START_DETAILS and PIN_FLD_*_END_DETAILS fields in the **/deal** object's product and discount arrays. For information about the values stored in the details fields, see "Storing Start and End Times for Products and Discounts in a Deal" for more information.

Specifying Relative Start and End Times for Products and Discounts in a Deal

When the purchase, cycle, or usage period of a product or discount has a relative start or end time, the details about the relative period are passed in opcode flists in UNIT and OFFSET fields.

The following UNIT fields specify the type of relative unit:

- PIN_FLD_PURCHASE_START_UNIT
- PIN_FLD_PURCHASE_END_UNIT
- PIN_FLD_CYCLE_START_UNIT
- PIN_FLD_CYCLE_END_UNIT
- PIN_FLD_USAGE_START_UNIT
- PIN_FLD_USAGE_END_UNIT

The unit can be one of these values:

- Seconds = 1
- Minutes = 2
- Hours = 3
- Days = 4
- Months = 5
- Accounting cycles = 8

The following OFFSET fields specify the number of units in the relative period:

- PIN_FLD_PURCHASE_START_OFFSET
- PIN_FLD_PURCHASE_END_OFFSET
- PIN_FLD_CYCLE_START_OFFSET
- PIN_FLD_CYCLE_END_OFFSET
- PIN_FLD_USAGE_START_OFFSET
- PIN_FLD_USAGE_END_OFFSET

The UNIT and OFFSET fields for each purchase, cycle, and usage period are used in combination to determine the relative offset period. For example:

- If a product's PIN_FLD_CYCLE_START_UNIT has a value of **8** (accounting cycles) and PIN_FLD_CYCLE_START_OFFSET has a value of **3**, the cycle fee starts three accounting cycles after the product is purchased.
- If a discount's PIN_FLD_PURCHASE_END_UNIT has a value of **8** (accounting cycles) and PIN_FLD_PURCHASE_END_OFFSET has a value of **3**, the discount expires three accounting cycles after it is activated.

You can specify any number of units in the OFFSET fields, up to 1048576. This is equivalent to approximately 12 days in seconds, 728 days in minutes, and 119 years in hours.

Information about the relative offset validity periods is stored in the BRM database in DETAILS fields in the **/deal** object's products and discounts arrays. For more information, see "Storing Start and End Times for Products and Discounts in a Deal".

The product and discount start and end dates (the PIN_FLD_*_START_T and PIN_FLD_*_END_T fields) are not set in the **/deal** object but are set in the **/purchased_product** and **/purchased_discount** objects when the product or discount is purchased.

Storing Start and End Times for Products and Discounts in a Deal

When a product or discount has a relative start or end time, the relative information passed in the PIN_FLD_*_UNIT and PIN_FLD_*_OFFSET fields of opcode flists is stored in the database in PIN_FLD_*_DETAILS fields in the **/deal** object:

- PIN_FLD_PURCHASE_START_DETAILS
- PIN_FLD_PURCHASE_END_DETAILS
- PIN_FLD_CYCLE_START_DETAILS
- PIN_FLD_CYCLE_END_DETAILS
- PIN_FLD_USAGE_START_DETAILS
- PIN_FLD_USAGE_END_DETAILS

The START-DETAILS and END-DETAILS fields are 32-bit integer fields that store three values: the mode of the validity period, the relative offset unit, and the number of offset units in the relative period:

- **Mode**: The mode specifies generally when the validity period starts or ends. The mode also helps to determine how the start and end dates are set in the account's **/purchased_product** and **/purchased_discount** objects when the product or discount is purchased. The mode is stored in the lower 8th bit and can have one of the following values:

  For the START-DETAILS:

  - **1** = PIN_VALIDITY_IMMEDIATE: The validity period starts when the product or discount is purchased. When purchased, the PIN_FLD_*_START_T field in the account's **/purchased_product** or **/purchased_discount** object is set to the purchase time.
  - **3** = PIN_VALIDITY_FIRST_USAGE: The validity period starts when the product or discount is first used to rate the customer's usage.
  - **4** = PIN_VALIDITY_RELATIVE: The validity period starts relative to the purchase date. When the product or discount is purchased, the start date is calculated by adding the relative offset period to the purchase date. The start date is then set in the purchased product's or purchased discount's PIN_FLD_*_START_T field.

  For the END-DETAILS:

  - **2** = PIN_VALIDITY_NEVER: The validity period never ends. When the product or discount is purchased, the purchased product's or purchased discount's PIN_FLD_*_END_T field is set to **0**.
  - **4** = PIN_VALIDITY_RELATIVE: The validity period ends relative to when it starts. When the product or discount is purchased, the end date is calculated by adding the relative offset period to the start date. The end date is then set in the purchased product's or purchased discount's PIN_FLD_*_END_T field.

When PIN_FLD_*_START_T and PIN_FLD_*_END_T are set in the account's **/purchased_product** and **/purchased_discount** objects, the mode in the details fields of those objects is set to PIN_VALIDITY_ABSOLUTE.

- **Relative offset unit**: This value is set when the product or discount starts at a time relative to the purchase date or ends at a time relative to the start date. It specifies the type of offset unit and corresponds to the value of PIN_FLD_*_START_UNIT or PIN_FLD_*_END_UNIT that is passed in opcode flists. The relative offset unit is stored in the next four bits of the details field and can have one of the following values:
  - Seconds = 1
  - Minutes = 2
  - Hours = 3
  - Days = 4
  - Months = 5
  - Accounting cycles = 8
- **Number of offset units**: This value specifies how many offset units are in the relative period and corresponds to the value of PIN_FLD_*_START_OFFEST or PIN_FLD_*_END_OFFSET that is passed in opcode flists. The number of offset units is stored in the remaining 20 bits of the details field.

Customizing How to Create and Delete Deals

To customize how to create and delete deals, use the following policy opcodes:

- To enable data modification during **/deal** object creation, use the PCM_OP_PRICE_POL_PREP_DEAL policy opcode.

  By default, this policy opcode is an empty hook, but you can customize it to enhance **/deal** objects with additional data not provided either by the GUI application or by the Price List FM or to perform other custom actions.

  This policy opcode is called before the Price List FM performs any validation checks.
- To enable validation during **/deal** object creation, use the PCM_OP_PRICE_POL_VALID_DEAL policy opcode. This policy opcode can be used to perform validations in addition to those performed by the Price List FM.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_DEAL. By default, this policy opcode is an empty hook; it returns the result PIN_PRICE_VERIFY_PASSED.
- To verify that deleting a **/deal** object is permitted, use the PCM_OP_PRICE_POL_DELETE_DEAL policy opcode. Use this opcode to perform validations in addition to those performed by the Price List FM.

  PCM_OP_PRICE_COMMIT_DEAL calls this policy opcode before deleting the **/deal** object. By default, this policy opcode is an empty hook provided to facilitate customization. It returns the result PIN_PRICE_VERIFY_PASSED.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.

Managing /plan Objects

Use the PCM_OP_PRICE_COMMIT_PLAN opcode to create, modify, or delete **/plan** objects in the BRM database.

This opcode accepts an flist consisting of a PIN_FLD_PLAN array, each element of which represents a **/plan** object.

PCM_OP_PRICE_COMMIT_PLAN determines whether to create, modify, or delete the specified **/plan** object:

- When the object does not exist, this opcode *creates* the specified **/plan** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_PREP_DISCOUNT policy opcode to perform any custom preparation. By default, this policy opcode is an empty hook, but you can customize it to enhance **/plan** objects with additional data or to perform other custom actions.
  - Calling the PCM_OP_PRICE_POL_VALID_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate any product specification attributes defined for **/plan** objects. See "Defining Product Specification Attributes for Pricing Components" for more information.
  - Creating the specified object in the BRM database.
- When the object already exists, this opcode *modifies* the specified **/plan** object in the BRM database.

> **Note:**
>
> This opcode overwrites data in existing **/plan** objects, so be sure you pass in the correct object to modify.

- When the PIN_FLD_DELETED_FLAG is set, this opcode *deletes* the specified **/plan** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_DELETE_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Deleting the specified object in the BRM database.

PCM_OP_PRICE_COMMIT_PLAN performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_PLAN sets PIN_FLD_RESULTS to **1**.
- When any operation fails, PCM_OP_PRICE_COMMIT_PLAN_LIST sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Managing /dependency Objects

Use the PCM_OP_PRICE_COMMIT_DEPENDENCY opcode to create, modify, or delete a **/dependency** object. This object specifies any dependencies between two discounts or between a discount and a plan.

This opcode accepts an flist that includes a PIN_FLD_DEPENDENCY array, each element of which represent a **/dependency** object.

Flist /dependency Arrays

The arrays defining prerequisite and mutually exclusive relationships require the names of the two product objects. The name of the product that *requires* another product is listed in the PIN_FLD_DEPENDENT_NAME field; the name of the *required* product is listed in the PIN_FLD_DEPENDEE_NAME field. For example, if your business has an email product that requires a Basic Dialup product, **email** is the dependee and **Basic Dialup** is the dependent:

```
0 PIN_FLD_DEPENDENCIES   ARRAY  [12]
1  PIN_FLD_DEPENDEE_OBJ    POID [0] 0.0.0.1 /deal -1 0
1  PIN_FLD_DEPENDENT_OBJ   POID [0] 0.0.0.1 /deal -1 0
1  PIN_FLD_DEPENDEE_NAME    STR [0] "Basic Dialup"
1  PIN_FLD_DEPENDENT_NAME   STR [0] "email"
1  PIN_FLD_TYPE             ENUM [0] 1
```

PCM_OP_PRICE_COMMIT_DEPENDENCY performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_DEPENDENCY sets PIN_FLD_RESULTS to **1**.
- When any operation fails, PCM_OP_PRICE_COMMIT_DEPENDENCY sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Customizing How to Create and Delete Dependencies

To customize how to create and delete dependencies, use the following policy opcodes:

- To enable data modification during **/dependency** object creation, use the PCM_OP_PRICE_POL_PREP_DEPENDENCY policy opcode. Use this policy opcode to enhance **/dependency** objects with additional data not provided by either the GUI application or by the Price List FM. For example, you would modify this policy opcode if your business requires that you change a mutually exclusive relationship into a required one, without using Pricing Center.

  PCM_OP_PRICE_COMMIT_DEPENDENCY calls this policy opcode before creating a new **/dependency** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  This policy opcode is called before the Price List FM performs any validation checks.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.

- To enable validation during **/dependency** object creation, use the PCM_OP_PRICE_POL_VALID_DEPENDENCY policy opcode. This policy opcode can be used to change **/dependency** relationships without using Pricing Center.

  This policy opcode is called by PCM_OP_PRICE_SET_PRICE_LIST and PCM_OP_PRICE_COMMIT_DEPENDENCY before creating a **/dependency** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.

- To verify that deleting a **/dependency** object is permitted, use the PCM_OP_PRICE_POL_DELETE_DEPENDENCY policy opcode. Use this policy opcode to perform validations in addition to those performed by the Price List FM.

  For example, you can use this policy opcode to confirm that the valid time period for the **/dependency** object has expired before allowing it to be deleted.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_DEPENDENCY before it deletes a **/dependency** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.

Managing /transition Objects

Use the PCM_OP_PRICE_COMMIT_TRANSITION opcode to create, modify, or delete a **/transition** object.

This opcode accepts an flist consisting of a PIN_FLD_TRANSITION array, each element of which represents a **/transition** object.

PCM_OP_PRICE_COMMIT_TRANSITION determines whether to create, modify, or delete the specified **/transition** object:

- When the object does not exist, this opcode *creates* the specified **/transition** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_PREP_TRANSITION policy opcode to perform any custom preparation. By default, this policy opcode is an empty hook, but you can customize it to enhance **/transition** objects with additional data or to perform other custom actions.
  - Calling the PCM_OP_PRICE_POL_VALID_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Creating the specified object in the BRM database.
- When the object already exists, this opcode *modifies* the specified **/transition** object in the BRM database.

  > **Note:**
  >
  > This opcode overwrites data in existing **/transition** objects, so be sure you pass in the correct object to modify.

- When the PIN_FLD_DELETED_FLAG is set, this opcode *deletes* the specified **/transition** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_DELETE_TRANSITION policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Deleting the specified object in the BRM database.

PCM_OP_PRICE_COMMIT_TRANSITION performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_TRANSITION sets PIN_FLD_RESULTS to **1**.
- When any operation fails, PCM_OP_PRICE_COMMIT_TRANSITION sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Customizing How to Create and Delete Transitions

To customize how to create and delete transitions, use the following policy opcodes:

- To enable data modification during **/transition** object creation, use the PCM_OP_PRICE_POL_PREP_TRANSITION policy opcode. Use this policy opcode to enhance **/transition** objects with additional data not provided by either the GUI application or by the Price List FM.

  For example, if your business waives purchase and cancellation fees for its San Francisco customers, you would implement that logic in this policy opcode.

  PCM_OP_PRICE_COMMIT_TRANSITION calls this policy opcode before creating a new **/transition** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  This policy opcode is called before the Price List FM performs any validation checks.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.
- To enable validation during **/transition** object creation, use the PCM_OP_PRICE_POL_VALID_TRANSITION policy opcode. This policy opcode can be used to change **/transition** relationships without using Pricing Center.

  For example, if your business requires that you prohibit underage customers from transitioning to a specific deal, you would add that limitation to this policy opcode.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_TRANSITION before creating a **/transition** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  This policy opcode is called before the Price List FM creates the **/transition** object.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.
- To verify that deleting a **/transition** object is permitted, use the PCM_OP_PRICE_POL_DELETE_TRANSITION policy opcode. Use this policy opcode to perform validations in addition to those performed by the Price List FM.

  For example, you can use this policy opcode to confirm that the valid time period for the **/transition** object has expired before allowing it to be deleted.

  This policy opcode is called by PCM_OP_PRICE_COMMIT_TRANSITION before it deletes a **/transition** object. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.

  Success or failure is indicated by the value returned for PIN_FLD_RESULTS. When successful, PIN_FLD_RESULTS is set to **1**. When any operation fails, PIN_FLD_RESULTS is set to **0**.

Managing /group/plan_list Objects

Use the PCM_OP_PRICE_COMMIT_PLAN_LIST opcode to commit **/group/plan_list** objects to the BRM database.

This opcode connects to the database, opens a transaction, and retrieves the **/plan_list** flist. PCM_OP_PRICE_COMMIT_PLAN_LIST searches the PIN_FLD_PLAN flist to match the name and type fields in the plan list flist. This plan is checked against existing plans in the database. If the plan does not exist, this opcode reports an error and stops the transaction. If the plan does exist, this opcode adds the flist to the PIN_FLD_RESULTS array. This process is repeated until all plan list entries are added to the PIN_FLD_RESULTS array and committed to the database. The committed changes include new, modified, or deleted objects.

PCM_OP_PRICE_COMMIT_PLAN_LIST performs all operations within a single transaction, so success or failure is indicated by the value returned for PIN_FLD_RESULTS:

- When successful, PCM_OP_PRICE_COMMIT_PLAN_LIST sets PIN_FLD_RESULTS to **1**.
- When any operation fails, PCM_OP_PRICE_COMMIT_PLAN_LIST sets PIN_FLD_RESULTS to **0**. It also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Managing /sponsorship Objects

Use the PCM_OP_PRICE_COMMIT_SPONSORSHIP opcode to create, change, or delete a **/sponsorship** object.

PCM_OP_PRICE_COMMIT_SPONSORSHIP determines whether to create, modify, or delete the specified **/sponsorship** object:

- When the object does not exist, this opcode *creates* the specified **/sponsorship** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_PREP_DISCOUNT policy opcode to perform any custom preparation. By default, this policy opcode is an empty hook, but you can customize it to enhance **/sponsorship** objects with additional data or to perform other custom actions.
  - Calling the PCM_OP_PRICE_POL_VALID_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Calling the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING policy opcode. By default, this policy opcode is an empty hook, but you can customize it to validate any product specification attributes defined for **/sponsorship** objects. See "Defining Product Specification Attributes for Pricing Components" for more information.
  - Creating the specified object in the BRM database.
  - Preparing a list of all **/sponsorship** objects that were created.
  - Generating the **/event/notification/price/sponsorships/modify** notification event.
- When the object already exists, this opcode *modifies* the specified object by doing the following:
  - Updating the specified object in the BRM database.
  - Preparing a list of all **/sponsorship** objects that were modified.

  - Generating the **/event/notification/price/sponsorships/modify** notification event.

> **Note:**
>
> This opcode overwrites data in existing **/sponsorship** objects, so be sure you pass in the correct object to modify.

- When the PIN_FLD_DELETED_FLAG is set, this opcode *deletes* the specified **/sponsorship** object by doing the following:
  - Calling the PCM_OP_PRICE_POL_DELETE_DISCOUNT policy opcode to perform any custom validation. By default, this policy opcode is an empty hook, but you can customize it to validate field values or to perform other custom actions.
  - Deleting the specified object in the BRM database.

PCM_OP_PRICE_COMMIT_SPONSORSHIP performs all operations within a single transaction, so success or failure is indicated by the value returned in the PIN_FLD_RESULTS output flist field: a value of **1** indicates that all operations succeeded and a value of **0** indicates that the operation failed. If the operation fails, the opcode also returns two additional fields: PIN_FLD_RESULT_FORMAT, which lists all error codes, and PIN_FLD_DESCR, which provides a concise error description.

Managing Tax Selectors and Tax Exemption Selectors

Tax selectors enable you to apply tax codes to your customers' products based on specified attributes in **/account**, **/service**, **/event**, and **/profile** objects. Each tax selector specifies:

- The name of the tax selector.
- The name of the tax code along with its associated criteria; that is, the list of event, service, profile, or account fields that an event must contain to qualify for the tax code.

BRM stores information about tax selectors in **/tax_selector** objects, and tax exemptions in **/tax_exemption_selector** objects. For more information about these objects, see *BRM Storable Class Reference*.

To manage tax selectors, use the following opcodes:

- PCM_OP_PRICE_CREATE_SELECTOR. See "Creating Tax Selectors".
- PCM_OP_PRICE_GET_SELECTOR. See "Retrieving Tax Selectors".
- PCM_OP_RATE_EVALUATE_SELECTOR. See "Retrieving the Tax Code from a Selector".

Creating Tax Selectors

Use the PCM_OP_PRICE_CREATE_SELECTOR opcode to create a tax selector (a **/tax_selector** or **/tax_exemption_selector** object) in the BRM database. This opcode is called by PCM_OP_PRICE_SET_PRICE_LIST.

PCM_OP_PRICE_CREATE_SELECTOR performs the following operations:

1. Reads the selector rules and converts them into a buffer field.
2. If the PIN_FLD_POID input flist field is set to **0.0.0.***x* **/tax/selector -1 0**, does the following:
   a. Creates a **/tax_selector** object.

b. Adds the **/tax_selector** POID to the **/rate_plan** object.

c. Returns the POID of the new **/tax_selector** object in the opcode's output flist.

3. If the PIN_FLD_POID input flist field is set to **0.0.0.***x* **/tax_exemption_selector -1 0**, does the following:

a. Creates a **/tax_exemption_selector** object.

b. Adds the **/tax_exemption_selector** POID to the **/product** object.

c. Returns the POID of the new **/tax_exemption_selector** object in the opcode's output flist.

Retrieving Tax Selectors

Use the PCM_OP_PRICE_GET_SELECTOR opcode to retrieve a tax selector (**/tax_selector** or **/tax_exemption_selector** object) from the BRM database. This opcode is called by PCM_OP_PRICE_GET_PRICE_LIST.

To retrieve a tax selector, pass one of the following in the opcode's input flist:

- PIN_FLD_POID set to a type-only POID and PIN_FLD_NAME set to the name of the tax selector. For example:

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /tax_selector -1 0
0 PIN_FLD_NAME     STR [0] "my_tax_selector"
```

- PIN_FLD_POID set to the tax selector object's POID. For example:

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /tax_exemption_selector 120916 0
```

The opcode returns the tax selector object POID along with details about the tax selector, such as the tax selector rules and tax codes.

Retrieving the Tax Code from a Selector

Use the PCM_OP_RATE_EVALUALTE_SELECTOR opcode to determine the correct tax code or tax exemption code to apply based upon the attributes provided. This opcode is called by PCM_OP_PRICE_GET_PRICE_LIST.

To determine the correct tax code or tax exemption code to apply, pass one of the following in the opcode's input flist:

- If pipeline is enabled, PIN_FLD_POID set to a type-only POID. For example:

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /tax_selector -1 0
```

- PIN_FLD_POID set to the tax selector object's POID. For example:

```
0 PIN_FLD_POID    POID [0] 0.0.0.1 /tax_exemption_selector 120916 0
```

PCM_OP_RATE_EVALUATE_SELECTOR performs the following operations:

1. Reads the event details and tax selector, and then converts them into a buffer field.
2. Compares the event details to the criteria for each tax code. If the event matches the event, service, profile, or account fields required for a tax code, the tax code is added to the output flist.

3. Calls PCM_OP_RATE_POL_POST_EVALUATE_SELECTOR to perform optional customization.
4. Returns the tax code or tax exemption code.

Managing Filter Sets

Filter sets enable you to apply system products and system discounts to a select group of customers (for example, those with the best credit history). You define which customers and events qualify for specified products and discounts by using Pricing Center or a custom client application. Each filter set specifies:

- The criteria an EDR must meet to qualify for the filter set: that is, list of required EDR fields and their values
- The list of applicable system products and system discounts, along with their validity dates and priorities

BRM stores information about each filter set in **/filter_set/product** objects.

To manage filter sets, use the following opcodes:

- PCM_OP_FILTER_SET_CREATE. See "Creating Filter Sets".
- PCM_OP_FILTER_SET_UPDATE. See "Updating Filter Sets".
- PCM_OP_FILTER_SET_DELETE. See "Deleting Filter Sets".

Creating Filter Sets

Use the PCM_OP_FILTER_SET_CREATE opcode to create **/filter_set/product** objects in the BRM database. This opcode is called directly by Pricing Center.

PCM_OP_FILTER_SET_CREATE performs the following operations:

1. Confirms that the object to be created is a **/filter_set/product** object, or one subclassed from it. The type of object is specified in the PIN_FLD_POID field of the input flist.
2. Creates the **/filter_set/product** object.
3. Generates an **/event/filter_set/create** object to record details of the event.
4. Returns the POID of the new **/filter_set/product** object.

Updating Filter Sets

Use the PCM_OP_FILTER_SET_UPDATE opcode to modify existing **/filter_set/product** objects. This opcode is called directly by Pricing Center.

PCM_OP_FILTER_SET_UPDATE performs the following operations:

1. Determines whether to add, modify, or delete data in the object by checking the PIN_FLD_FLAG field:
   - When the flag is set to **0**, the opcode *modifies* filter set data.
   - When the flag is set to **1**, the opcode *adds* filter set data.
   - When the flag is set to **2**, the opcode *deletes* filter set data.
2. Modifies data in the specified **/filter_set/product** object.
3. Creates an **/event/filter_set/update** object to record details of the event.

4. Returns the POID of the **/filter_set/product** object.

Deleting Filter Sets

Use the PCM_OP_FILTER_SET_DELETE opcode to delete **/filter_set/product** objects from the BRM database. This opcode is called directly by Pricing Center.

PCM_OP_FILTER_SET_DELETE performs the following operations:

1. Deletes the specified **/filter_set/product** object.
2. Generates an **/event/filter_set/delete** object to record details of the event.
3. Returns the POID of the **/filter_set/product** object.

23
Testing Your Price List

This chapter describes the procedure for testing an Oracle Communications Billing and Revenue Management (BRM) price list. Testing is an important step to take before using the price list for customers.

Topics in this document:

- About Using a Test Database
- Creating Test Accounts
- Generating Test Account Activity
- Advancing the Date
- Generating a Test Invoice
- Example Price List Test.

For information about creating a price list, see "About Creating a Price List".

About Using a Test Database

Before testing a price list, set up a test database that is separate from your production database. Because you simulate account activity and time changes when you test your price list, you would create erroneous events in a production BRM database that contains real customer accounts.

> **Note:**
>
> You can remove all price list elements in a database by using the **loadpricelist utility -p** parameter. See "loadpricelist".

Creating Test Accounts

Use Customer Center to create test accounts in your test database. Note the login name for each account you create, because you will need it later.

> **Note:**
>
> To create accounts with your new plans, you must add them to a plan list.

When selecting a payment method, you can select either Invoice or Credit Card. If you select Credit Card, also do the following:

- Ensure that the necessary Data Managers and applications are properly configured and running:

  - For Paymentech, **dm_fusa**, **answer_b**, and **answer_s** should be running.

- You must use one of the following pairs of credit card numbers and expiration dates listed in Table 23-1 for your test accounts:

Table 23-1 Example Credit Card Expiration Data

Credit Card Number	Expiration Date
4444 1111 2222 3333	0999
4101 0000 0000 0000	any expiration date

> **Note:**
>
> A simple cycle rate, such as a monthly fee for a service, should appear in an account's balance as soon as you create the account.

Generating Test Account Activity

Use the **sample_act** utility to generate activity for a test account. You use this utility to simulate a customer using a product or service without the product or service actually being available. You can then see if the activity changes the test account's balance as you expect.

Perform these tasks to generate activity with **sample_act**:

- Running sample_act
- Checking Results of Running sample_act

Running sample_act

1. Go to *BRM_Home*/**source/apps/sample**.
2. Edit the **pin.conf** file for the **cm_ptr** to point to your CM process. Also, edit the login type, login name, and password entries.
3. Run the **sample_act** utility for each account you want to include in your test.

   The syntax for **sample_act** depends on the type of event you are simulating:

   - **Session event**: A usage event, such as hours of Internet access. If the rate depends on the passage of time, use a session event.

     For a session event, use this syntax:

     ```
     sample_act -v -e session -l login -d duration -s servicename
     ```

   - **Activity event**: An item or cycle fee that does not depend on time, such as a monthly fee.

     For an activity event, use this syntax:

     ```
     sample_act -v -e activity -l login -q quantity -s servicename
     ```

   Following is a description of the most common **sample_act** options.

   - **-v**: Use to run **sample_act** in verbose mode. In this mode, you see status messages at the command prompt, including messages that the event generated successfully or that an error occurred. There is no argument.

- [**-n RUM** *Name* **-q** *quantity* [**-u** *Unit*]]
- [**-b** *event time*] (as MM:DD:YY:HH:MM:SS or unixtime)
- [**-A** *ANI*] [**-D** *DNIS*] [**-Y** *call_type*]*:* ANI is the source telephone number and DNIS is the destination telephone number
- [**-X** *int_indicator*] [**-O** *origin_gw*]
- [**-N** *new Timezone ID String*]
- [**-I** *termserv_id*] [**-P** *termserv_port*] [**-T** *old timezone offset*]
- **-e** *activity*|*session* [**-c** *custom_subtype*]: Enter activity or session.
- **-l** *login*: Enter the login name for the test account and service for which you want to generate activity.
- **-q** *quantity*: For an activity event, enter the quantity purchased.
- **-d** *duration*: For a session event, enter the duration of the simulated event in seconds. The default is 3600 seconds, or one hour.
- **-s** *service_type* **-l** *service_login*: Enter the name of the service for which you are simulating the event, such as **/service/ip**.

The following example simulates using three hours of Internet service. The login is **pin_user** and the service is **/service/ip**.

```
% sample_act -v -e session -l pin_user -d 10800 -s /service/ip
```

To test a rate that changes at different times, such as an Internet service rate with peak and off-peak rates, you can advance the system date in conjunction with running **sample_act**. See "Advancing the Date ".

Before advancing the time, you must edit the **sample_act** configuration file so that **sample_act** responds to the changed time. See "Setting Up the pin_virtual_time Utility".

Checking Results of Running sample_act

To get the results of your test after running **sample_act**:

1. Find the test accounts in Customer Center.
2. Click the **Balance** tab.
3. For detailed information about balance changes, use Event Browser.

Advancing the Date

To properly test a price list, you must simulate the passage of time. For example, you should test:

- If accounts are billed correctly. Advance the date to the next billing cycle, then run billing and generate an invoice.
- If a time-based discount takes effect correctly. If you have a deal that includes 90 days of free email service, for example, advance the date more than 90 days to see if the account starts being charged for the service.
- If folds are working correctly. If a product includes 10 free hours of Internet service per month, for example, advance the date by a month or more to ensure that unused free hours do not carry over.

The **pin_virtual_time** utility enables you to simulate changes in the BRM system time.

> **Note:**
>
> If you use **pin_virtual_time** with a BRM system that includes Pipeline Manager, you must set the **VirtualTime** parameter for the DAT_BalanceBatch module to **True** to ensure that balances are calculated correctly.

> **Note:**
>
> The **pin_virtual_time** utility works only on a single computer. Typically, you set up a test BRM system on just one computer. If you run **pin_virtual_time** on a system that is distributed across multiple computers, you must carefully coordinate time changes on all the computers.

Perform the following tasks to advance the date:

- Setting Up the pin_virtual_time Utility
- Running pin_virtual_time
- Stopping and Starting BRM

Setting Up the pin_virtual_time Utility

1. Go to the *BRM_home***/sys/test** directory.
2. Run this command to create a file that **pin_virtual_time** requires, called **pin_virtual_time_file**:

   ```
   pin_virtual_time -m 0 -f BRM_Home/bin/pin_virtual_time_file
   ```

   > **Note:**
   >
   > *BRM_home***/lib/pin_virtual_time_file** is the standard path and file name, but you can change it.

3. Add the following entry to the configuration file in the *BRM_Home***/sys/test** directory:

   ```
   - - pin_virtual_time  pin_virtual_time_file
   ```

   Replace *pin_virtual_time_file* with the path and name of the mapped file created in the previous step.

   Adding the entry to the configuration file in **sys/test** enables you to run **pin_virtual_time** from that directory. The directory from which you run **pin_virtual_time** must contain a configuration file with the **pin_virtual_time** entry.

4. Add the entry in step 3 to the configuration file for each program you want to respond to an altered time.

   To test your price list, edit the configuration files for at least the applications listed in Table 23-2:

Table 23-2 Configuration Files for Testing Applications

Application	Configuration File Location
CM	**sys/cm/**
Oracle DM	**sys/dm_oracle/**
Billing and invoice utilities	**apps/pin_billd/**
sample_act utility	**source/apps/sample**

To ensure that all applications respond to **pin_virtual_time** changes, add the entry to all the configuration files in your test BRM system.

Entry example:

```
- - pin_virtual_time BRM_home/bin/pin_virtual_time_file
```

Running pin_virtual_time

After you set up the **pin_virtual_time** utility, run **pin_virtual_time** to advance the BRM date and time.

Syntax for **pin_virtual_time**:

```
pin_virtual_time -m 2 MMDDHHMM[CC]YY.[SS]
```

For the string MMDDHHMM[CC]YY.[SS], enter the date and time you want BRM to use in this format: month, date, hour, minute, year, and seconds. You must enter at least two digits for the year, but you can enter four. Seconds are optional.

For example, to set the date and time to 9/3/99 and 11:30, respectively:

```
% pin_virtual_time -m 2 090311301999.00
```

The command displays this message at the command prompt:

```
filename BRM_Home/lib/pin_virtual_time_file, mode 2, time: Fri Sept 03 11:30:00 1999
```

The time then advances normally from the new reset time.

Stopping and Starting BRM

After you run **pin_virtual_time**, you must stop and restart BRM to read in the new time:

1. Log on to the computer that is the server for your test BRM system. Usually, your test system should be on a single machine.

   > **Note:**
   >
   > Log in as something other than **root**.

2. Exit all BRM client tools, such as Customer Center and Pricing Center.
3. Stop and restart the CM and Oracle Data Manager.

Generating a Test Invoice

Generate invoices for your test accounts to see if the accounts are billed correctly. Before following this procedure, you should have generated account activity and advanced the date to the beginning of the next billing cycle. Go to the **Payment** tab in Customer Center to find the Next Billing Start date for an account.

To run billing and generate a test invoice:

1. Go to the *BRM_Home***/apps/pin_billd** directory.
2. At the command prompt, enter the following:

   ```
   % pin_bill_day
   ```

   This command is a script that runs several billing utilities.

   BRM creates an invoice for each active account.
3. Open a test account in Customer Center and go to the **Payment** tab.
4. Ensure that the Current Billing Start and End dates are what you expect.
5. Search for the invoice for the current account, and see that it contains the results of your test. See "Displaying invoices" in *BRM Designing and Generating Invoices*.

Example Price List Test

This example takes you through the steps for testing a plan that includes the following:

- A monthly Internet access fee of $9.95.
- An hourly fee of $2.00 for prime-time Internet service usage. Prime time is 8:00 a.m. to 6:00 p.m. weekdays.
- An hourly fee of $1.00 for off-peak Internet service usage. This includes nights and weekends.

You can test this plan as follows:

1. In Customer Center, create a test account by using the plan you are testing.
2. Go to the **Balance** tab. The current balance for the US Dollar resource should be $9.95, because the monthly access fee impacts the balance right away.
3. Close the test account.
4. Set up the **pin_virtual_time** utility, by using the procedure in "Setting Up the pin_virtual_time Utility". Ensure that you at least add the **pin_virtual_time** entry to the configuration files for the Connection Manager, Oracle Data Manager, and **sample_act** utility.
5. Go to the *BRM_Home***/sys/test** directory.
6. Change the BRM system time so it is close to the time when one rate ends and the other begins. For example, set the time to 6:00 a.m. the following day.

   If you created the account on May 1, 1999, you would run the following command to change BRM's system time to 6:00 a.m. the next morning:

   ```
   pin_virtual_time -m 2 0502060099 -l pin_user
   ```

   In the above command, *pin_user* is the login assigned to the test account.

7. Stop and restart BRM, by using the procedure in "Stopping and Starting BRM".
8. Go to *BRM_Home* **/source/apps/sample**.
9. Generate four hours of Internet access activity for the account by running this command:

   ```
   sample_act -v -e session -r ip/dialup/async/hourly -s /service/ip -d 14400 -l
   pin_user
   ```

   In the above command, **ip/dialup/async/hourly** is the rate category, **/service/ip** is the service, 14400 is four hours in seconds, and **pin_user** is the login name for the account.
10. Reopen the account in Customer Center.
11. Go to the **Balance** tab.

    The current balance for the US Dollar resource should now be $15.95. The **sample_act** command should have generated two hours of Internet access at the off-peak fee of $1 and two hours at the prime-time fee of $2. If you set up the rates correctly, this adds $6.00 to the previous balance of $9.95.

24

Using the XML Pricing Interface to Create a Price List

This chapter describes how to create and edit an Oracle Communications Billing and Revenue Management (BRM) price list by using the XML Pricing Interface.

Topics in this document:

About Creating and Modifying Price List Files

You create and modify price lists by using the following:

- **Pricing Center**: Pricing Center is a GUI application that provides on-screen wizards to help you create your price list. You create and modify the price list and retrieve and save it to the database all directly in Pricing Center. For more information, see Pricing Center Help.
- **XML Pricing Interface**: The XML Pricing Interface consists of the "loadpricelist" utility. You create price lists directly in an XML editor or text editor and then use the utility to load, retrieve, and delete the price list from the BRM database. For information on creating price lists with the XML Pricing Interface, see "About the XML Pricing Interface".

About the XML Pricing Interface

The XML Pricing Interface is a command-line utility called **loadpricelist** that you can use to download, edit, and load price lists.

> **Note:**
>
> The XML price list must follow the format detailed in the XML SchemaDefinition (XSD) file **price_list.xsd**. By default, the **price_list.xsd** file is installed in *BRM_Home*/**PricingCenter** and/or *BRM_Home*/**setup/scripts**.

About Downloading and Loading XML Price List Files

You can use the **loadpricelist** utility to download and load an entire price list or a partial price list. When you download a price list from the database, the utility writes the price list information you specify into an XML file. If you specify to download to an existing price list file, the utility overwrites the entire XML file.

When you load a price list from an XML file into the database, the utility modifies only those objects that have changed:

- If an object has changed, the utility updates the object's attributes in the database.
- If an object does not exist in the database, the utility creates the database object.
- If an object has not changed and is already in the database, the utility does not modify the database object.

Prerequisites for the XML Pricing Interface

Before you use the XML Pricing Interface, you must configure the following:

- Include the **loadpricelist.jar**, **xerces.jar**, **pcm.jar**, and **pcmext.jar** files in your CLASSPATH. The **xerces.jar**, **pcm.jar**, and **pcmext.jar** files are located in the **C:\Program Files\Common Files\Portal Software** directory and subdirectories by default.

  > **Note:**
  >
  > Instead of including the JAR files in your CLASSPATH, you can include them in the **loadpricelist** file.

- Include the location of the **Infranet.properties** file in your CLASSPATH.

  The default **Infranet.properties** file is in the **\common** directory. To put the **Infranet.properties** file in the local directory with the **loadpricelist** utility, add **./** (period-slash) to the beginning of the **Infranet.properties** CLASSPATH.

- Ensures that the **Infranet.properties** file specifies the correct login information for the BRM database you want to connect to.

  > **Note:**
  >
  > You might need to change this information if, for example, you want to download a price list from one BRM database, modify it, and load it to a different BRM database.

- Install Pricing Center or copy the **price_list.xsd** file to your system. You can find the **price_list.xsd** file on any BRM server in either the *BRM_Home***/PricingCenter** directory or the *BRM_Home***/setup/scripts** directory.

Starting the XML Pricing Interface

To start the XML Pricing Interface, at the command prompt, enter **loadpricelist** followed by an interactive or non-interactive command.

Getting Help for the XML Pricing Interface

To get help for the XML Pricing Interface:

1. Start the XML Pricing Interface, if necessary.
2. Enter **help** at the command prompt and press **Enter**.

Working in Interactive and Non-Interactive Mode

You can work in interactive and non-interactive mode in the XML Pricing Interface.

- In interactive mode, you issue a command for each action. For a list of commands that can be used in this mode, see "Syntax: Interactive Mode".
- In non-interactive mode, you can use special commands that batch several related actions together so the system performs them automatically. For a list of parameters that can be used in this mode, see "Syntax: Non-Interactive Mode".

You receive a confirmation message when the XML Pricing Interface carries out a command successfully. If you do not receive a confirmation message, look in the utility log file (**javapcm.log**) to find any errors. The log file is either in the directory where the utility was started or in a directory specified in the **Infranet.properties** file.

> **Note:**
>
> If an error occurs during a command, none of the command is carried out even if the error was only caused by part of the operation.

Creating Price Lists with the XML Pricing Interface

You create price lists in an XML editor or text editor and then load it into the BRM database by using the "loadpricelist" utility.

To create a price list:

1. Plan your price list. You should know in advance how you will set up your rates, products, deals, and plans. Ensure that the following pricing objects and data have been created in the BRM database:
   - services
   - events
   - resources
   - currency exchange rates
   - G/L IDs
   - tax codes and tax suppliers
   - ratable usage metrics (RUMs)
   - product-level provisioning categories
   - zones

   For information on planning your price list, see "About Creating a Price List".

2. Create an XML file, such as **price_list.xml**, in a text editor or XML editor.
3. Enter your price list information. Ensure that your price list follows the structure defined in the **price_list.xsd** file.

> **Note:**
>
> You can view sample XML price list files in the *BRM_Home***PricingCenter\Sample_Price_Plans** directory.

   See "XML Examples of Creating Pricing Components" for some pricing configuration examples using XML.
4. Save and close the file.
5. Ensure that any billable events in your price list are also included in the **pin_event_map** file. Otherwise, they will not be rated. See "Mapping Events and Services".
6. Load the price list into the BRM database. See "Loading a Price List into the Database with the XML Pricing Interface".

BRM can begin using the price list after it is committed to the database.

Modifying Existing Price Lists with the XML Pricing Interface

To modify an existing price list file:

1. Retrieve the existing price list file. The method you use depends on the file's location:
   - **The XML Pricing Interface**: Your price list file is saved in XML format. You can open it directly in an XML editor or text editor.
   - **The BRM database**: You must download the price list to your system and write the information to an XML file. See "Downloading Price Lists from the Database with the XML Pricing Interface".
2. Open the XML price list file in a text editor or XML editor.
3. Edit your price list file. Ensure that it follows the structure defined in the **price_list.xsd** file. By default, **price_list.xsd** is located in *BRM_Home***PricingCenter**.

> **Note:**
>
> You can view sample XML price list files in the *BRM_Home***PricingCenter\Sample_Price_Plans** directory.

4. Save and close the file.
5. Ensure that any billable events in your price list are also included in the **pin_event_map** file. Otherwise, they will not be rated. See "Mapping Events and Services".
6. Load the price list into the BRM database. See "Loading a Price List into the Database with the XML Pricing Interface".

BRM can begin using the price list after it is committed to the database.

Moving a Price List between Databases with the XML Pricing Interface

You can use the **loadpricelist** utility to move a price list from one database to another (for example, from a test database to a production database). You move the price list by downloading it from a *source* BRM database and then loading it into a *destination* BRM database.

To move a price list from one database to another:

1. Download the price list from the source database. See "Downloading Price Lists from the Database with the XML Pricing Interface".
2. If necessary, edit the XML file in a text editor or XML editor.
3. Load the price list into the destination BRM database. See "Loading a Price List into the Database with the XML Pricing Interface".

Downloading Price Lists from the Database with the XML Pricing Interface

You can download either the entire price list or just a subset of the price list from the BRM database into an XML file that can be edited.

> **Note:**
>
> Price lists are stored in the database in flist format. The utility converts the file into XML format as part of the download process.

Downloading a Full Price List

To download a full price list:

1. Ensure that the **Infranet.properties** file in your CLASSPATH has the correct login information for the BRM database you want to download price list information from. See "Setting global options" in *BRM Developer's Guide*.
2. Retrieve the price list from the BRM database and write it to an XML file.

   Interactive mode

   ```
   loadpricelist
   retrieve
   write [PriceListFile]
   ```

   where *PriceListFile* specifies the name and location of the file to which to extract the data.

> **Note:**
>
> You can retrieve only one price list at a time. If you retrieve another price list when one is already in internal memory, the existing internal flist is overwritten.

Non-interactive mode

```
loadpricelist [-f] -r PriceListFile
```

where *PriceListFile* specifies the name and location of the file to which to extract the data.

Your price list is written to the specified XML file. You can now edit it directly in an XML editor.

Downloading a Subset of the Price List

You can write specific pricing objects, such as **/product**, **/discount**, or **/sponsorship** objects, rather than the entire price list to the XML price list file.

To download a subset of the price list:

1. Ensure that the **Infranet.properties** file in your CLASSPATH has the correct login information for the BRM database you want to connect to.
2. Specify the pricing objects to retrieve according to their service type, modification time, or object type, and write them to a specified XML file.

   Interactive mode

   ```
   loadpricelist
   retrieve [-s ServiceType][-t ModifiedTime][product][discount][sponsorship]
   write PriceListFile
   ```

   where:

   - **-s** *ServiceType* specifies to retrieve the specified service type. You can list multiple service types by using a comma (**,**) as a delimiter.

     > **Note:**
     >
     > Do not use this parameter with the **sponsorship** parameter.

   - **-t** *ModifiedTime* retrieves objects that were modified after the specified timestamp.

     The time is specified in the ISO-8601 format: *YYYY*-*MM*-*DD***T***hh*:*mm*:*ss*TZD, with the server's local time zone as the default. For example:

     ```
     -t 1997-07-16T19:20:30+01:00
     ```

   - **product** specifies to retrieve only the **/product** objects from the database.
   - **discount** specifies to retrieve only the **/discount** objects from the database.
   - **sponsorship** specifies to retrieve only the **/sponsorship** objects from the database.
   - *PriceListFile* specifies the name and location of the file to which to extract the data.

   Non-interactive mode

   ```
   loadpricelist [-f] -r PriceListFile [-P][-D][-S][-s ServiceType][-t ModifiedTime]
   ```

where:

- **f** specifies to run the command without prompting for a confirmation.
- *PriceListFile* specifies the name and location of the file to which to extract the data.
- **-P** specifies to retrieve only the **/product** objects from the database.
- **-D** specifies to retrieve only the **/discount** objects from the database.
- **-S** specifies to retrieve only the **/sponsorship** objects from the database.
- **-s** *ServiceType* retrieves objects based on the specified service type. You can list multiple service types by using a comma (**,**) as a delimiter.

  > **Note:**
  >
  > Do not use this parameter with the **-S** parameter.

- **-t** *ModifiedTime* retrieves objects that were modified after the specified timestamp.

  The time is specified in the ISO-8601 format: *YYYY***-***MM***-***DD***T***hh***:***mm***:***ss*TZD, with the server's local time zone as the default. For example:

  ```
  -t 1997-07-16T19:20:30+01:00
  ```

Downloading Price Lists in Batches

If you have a large price list, downloading it by using **loadpricelist** can require more than the available system memory. In this situation, you should configure **loadpricelist** to retrieve and write pricelists in batches. When configured this way, **loadpricelist** performs a step search and writes the results into multiple XML files based on the batch size you specify.

You use the **infranet.batchsize** entry in the **loadpricelist Infranet.properties** file to configure the batch size. The batch size is expressed as the number of pricing objects per batch. For example, the following entry configures **loadpricelist** to retrieve and write pricing objects in batches of 500.

```
infranet.batchsize=500
```

When a batch size is configured, **loadpricelist** creates files based on the file name you specify in the command and on the type of pricing object, incrementing the name appropriately for each file created. For example, if the batch size is set to 500 and there are 1900 pricing objects in the database, then **loadpricelist** creates four XML files, the first three with 500 pricing objects and one with the remaining 400 objects.

When retrieving pricing objects in batches, you cannot choose which type of pricing object to retrieve; for example, only products, or only discounts. The -P, -D, -S, -s, and -t parameters are ignored.

In this scenario, this command:

```
loadpricelist -rf MyPriceList
```

results in the following four files being created (along with others for the other pricing objects in the database):

MyPriceList_product_1.xml

MyPriceList_product_2.xml

MyPriceList_product_3.xml

MyPriceList_product_4.xml

File naming follows the same pattern for all object types. For example, file names for **/discount** objects would start with **MyPriceList_discount_1.xml** and increment from there.

The **batchsize** entry is not included in the **loadpricelist** file by default. To set the batch size:

1. Open the **Infranet.properties** file for **loadpricelist** in a text editor.

   The default **Infranet.properties** file is in the **\common** directory, but may be located in the local directory with the **loadpricelist** utility**.**

2. Add the **batchsize** entry to the file using the following syntax:

   ```
   infranet.batchsize=batch_size
   ```

   where *batch_size* represents the number of pricing objects to include in each batch.

3. Save and close the file.

Loading a Price List into the Database with the XML Pricing Interface

After your price list is complete, you load it into the BRM database. The XML Pricing Interface automatically converts the XML price list into flist format when it is loaded into the database.

> **Note:**
>
> The price list file must be in the same directory as the **price_list.xsd** file. If they are not in the same directory, edit the **noNamespaceSchemaLocation** entry in the price list file to include the full path to **price_list.xsd**. By default, **price_list.xsd** is in the *BRM_Home***PricingCenter** directory. For example, if the XSD file is located in **C:\pricelists**, change the entry to:
> **xsi:noNamespaceSchemaLocation="C:\pricelists\price_list.xsd"**

To load a price list:

1. Ensure that the XML price list file is complete and follows the guidelines specified in the XSD file.
2. Ensure that the **Infranet.properties** file in your CLASSPATH has the correct login information for the BRM database you want to connect to.
3. Commit the price list XML file to the BRM database.

   Interactive mode

   ```
   loadpricelist
   read PriceListFile
   commit
   ```

   where *PriceListFile* specifies the name and location of the file to load into the database.

   Non-interactive mode

   ```
   loadpricelist [-f] -c PriceListFile
   ```

where:

- **f** specifies to run the command without prompting for a confirmation.
- *PriceListFile* specifies the name and location of the file to load into the database.

Purging Price Lists from the BRM Database

You can purge the following from the BRM database:

- All price list objects. See "Purging the Entire Price List from the BRM Database".
- A subset of the price list, based on the object type. See "Deleting a Subset of the Price List from the BRM Database".
- A specific price list object. See "Deleting Pricing Objects from the BRM Database".
- All dependency or transition objects. See "Purging Dependency or Transition Objects from the BRM Database".

Purging the Entire Price List from the BRM Database

To purge the entire price list from the database:

1. Ensure that the **Infranet.properties** file has the correct login information for the BRM database.
2. Purge the price list from the BRM database:

   Interactive mode

   ```
   loadpricelist
   purge
   ```

   Non-interactive mode

   ```
   loadpricelist [-f] -p
   ```

   where **f** specifies to run the command without prompting for a confirmation.

Deleting a Subset of the Price List from the BRM Database

> **Note:**
>
> To delete a subset of price list data from the BRM database, you must use interactive mode.

To delete a subset of the price list:

1. Ensure that the **Infranet.properties** file has the correct login information for the BRM database.
2. Start **loadpricelist** in interactive mode and read *PriceListFile* into internal memory.

   ```
   loadpricelist
   read PriceListFile
   ```

3. Specify the pricing data to delete:

```
delete [planlist[new|addon] | plan | bestpricing | deal | product | discount |
sponsorship]
```

where:

- **planlist** [**new|addon**] specifies to delete the plan list information from the BRM database. You can optionally specify to delete only the **new** plan lists or the **addon** plan lists.
- **plan** specifies to delete only the **/bestpricing** objects from the BRM database.
- **deal** specifies to delete only the **/deal** objects from the BRM database.
- **product** specifies to delete only the **/product** objects from the BRM database.
- **discount** specifies to delete only the **/discount** objects from the BRM database.
- **sponsorship** specifies to delete only the **/sponsorship** objects from the BRM database.

Deleting Pricing Objects from the BRM Database

> **Note:**
>
> To delete individual pricing objects from the BRM database, you must use interactive mode.

To delete pricing objects:

1. Ensure that the **Infranet.properties** file has the correct login information for the BRM database.
2. Start **loadpricelist** in interactive mode and read *PriceListFile* into internal memory.

   ```
   loadpricelist
   read PriceListFile
   ```

3. Specify the pricing object to delete.

   ```
   delete PriceObject
   ```

   where *PriceObject* specifies the pricing object to delete.

Purging Dependency or Transition Objects from the BRM Database

> **Note:**
>
> To delete only dependency or transition objects from the BRM database, you must use interactive mode.

To delete dependency or transition objects:

1. Ensure that the **Infranet.properties** file has the correct login information for the BRM database.
2. Specify the objects to delete.

- To delete only the dependency objects, enter:

```
loadpricelist
delete dependency
```

- To delete only the transition objects, enter:

```
loadpricelist
delete transition
```

25

XML Examples of Creating Pricing Components

This chapter describes how to configure pricing components in Oracle Communications Billing and Revenue Manager (BRM) by using an XML file and the **loadpricelist** utility.

Topics in this document:

The following topics show examples of creating pricing components using XML elements in your price list file.

Prorating Fees for Billing DOM Changes

When configuring rates in XML, you can specify how to apply cycle fees when customers change their billing day of the month (DOM) in the middle of their billing cycle. For example, on June 15, a customer changes his billing DOM from the 10th to the 30th, generating a partial month. You can specify whether the June 10 through June 30 bill contains the entire monthly cycle fee, a prorated monthly cycle fee, or no monthly cycle fee.

To do so, set **proration_flag** in the **<rate>** element to one of the following:

- **full**: Applies the total cycle fees.

- **prorate**: Prorates the cycle fees. This is the default.
- **none**: Applies no cycle fees.

The following sample XML entries specify to prorate cycle fees when customers change their billing DOM during a billing cycle:

```
<rate type="default" proration_flag="prorate">
   <description>$9.95 a month</description>
   <step_resource_id>840</step_resource_id>
   <quantity_tier>
      <balance_impact scaled_unit="none" flag="sponsorable discountable
proratable">
         <resource_id>840</resource_id>
         <glid>102</glid>
         <fixed_amount>10.0</fixed_amount>
         <scaled_amount>0</scaled_amount>
      </balance_impact>
   </quantity_tier>
</rate>
```

Setting Proration for Products in a Deal

When configuring deals in XML, you can specify how each product in the deal calculates prorated cycle fees. For example, a deal could contain one product or discount that prorates based on 30-day months and another that is based on the actual number of days in the month.

To set proration at the product level in your deals, set **flags** in the **<deal_product>** element to one of the following:

- **prorate_30_day**: Prorated cycle fees are calculated based on a 30-day month, regardless of the number of days in the billing cycle. For example, if a deal was owned for six days in a cycle, the prorated fee would be the cycle fee multiplied by 0.20 (6 ÷ 30).
- **prorate_days_in_month**: Prorated cycle fees are calculated based on the number of days in a particular month, such as 28 days in February, 31 days in March, and 30 days in April. For example, if a deal was owned for six days in March, the prorated fee would be the cycle fee multiplied by 0.19 (6 ÷ 31).
- **prorate_system**: Prorated cycle fees are calculated according to the system-wide setting in your CM **pin.conf** file. See "Enabling 30-Day-Based Proration" in *BRM Configuring and Running Billing*.

> **Note:**
>
> It is applicable only on products and does not affect discounts as discounts are always relative to products and therefore not subject to proration settings.

The following sample XML entries create a deal named **Sample Deal** with products **Product01** and **Product02**. **Product01** specifies to prorate all of its cycle fees using a 30-day month, **Product02** specifies to prorate all of its cycle fees based on the actual number of days in the month:

```
<deal customization_flag="optional" ondemand_billing="no">
```

```
<deal_name>Sample Deal</deal_name>
<description>My Sample Deal</description>
<permitted>/service/ip</permitted>
<deal_product flags="prorate_30_day">
   <product_name>Product01</product_name>
   <product_code>My Sample Product</product_code>
   <quantity>1.0</quantity>
   <purchase_discount>0.0</purchase_discount>
   <cycle_discount>0.0</cycle_discount>
   <usage_discount>0.0</usage_discount>
   <purchase_mode>0</purchase_mode>
<deal_product flags="prorate_days_in_month">
   <product_name>Product01</product_name>
   <product_code>Another Sample Product</product_code>
   <quantity>1.0</quantity>
   <purchase_discount>0.0</purchase_discount>
   <cycle_discount>0.0</cycle_discount>
   <usage_discount>0.0</usage_discount>
   <purchase_mode>0</purchase_mode>
</deal_product>
```

You can modify the **flags** value by using the PCM_OP_PRICE_SET_PRICE_LIST opcode. However, after customers subscribe to your deals, further updates to **flags** do not affect any purchased products.

You can override the **flags** value for a specific customer using the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL or PCM_OP_CUST_COMMIT_CUSTOMER opcode. See "Overriding Bundle Proration Settings at Customer Level" in *BRM Opcode Guide* for more information.

Setting Full Day Proration

To configure full day proration in XML, you use the <**validityRounding**> and <**scaleRounding**> elements.

<**validityRounding**> specifies whether to start the charge offer's validity period at the purchase time or midnight of the purchase day.

Set the <**validityRounding**> element to one of the following:

- **OFF**: Starts at the time of purchase.
- **ON**: Starts at midnight (00:00:00) of the day that the charge offer is purchased.
- **NOT_SET**: Uses your company's systemwide setting.

<**scaleRounding**> specifies how to calculate the scale.

Set the <**scaleRounding**> element to one of the following:

- **ON**: Calculate it based on full days.
- **OFF**: Calculate it based on the <**validityRounding**> setting.
- **NOT_SET**: Use the systemwide setting in the CM pin.conf file.

The following sample XML entries specify to start the validity period at midnight on the purchase day and charge for a full day on the first day of the billing cycle:

```
<?xml version="1.0" encoding="UTF-8"?>
<price_list version="7.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="price_list.xsd">
<product date_range_type="EVENT_DATE" partial="no" type="subscription">
    <product_name>MCF - $30</product_name>
    <product_code>MCF30</product_code>
    <offer_validity_rounding>NOT_SET</offer_validity_rounding>
    <scale_rounding>OFF</scale_rounding>
 </price_list>
```

Adding Pricing Tags to Rates

You can configure BRM to dynamically charge different one-time and recurring rates for the same event based on the date. For example, when a customer cancels a product, you could charge a default amount of $50, but charge $45 in June and $40 in July. To do so, you add a predefined pricing tag to the balance impacts in your price list XML file.

For more information about dynamic charging, see "Dynamically Changing One-Time and Recurring Fees Based On the Date".

To add pricing tags to rates, set the following elements under the **<balance_impact>** element:

- **<fixed_price_tag>**: Set this to the price tag name for overriding a fixed amount.
- **<scaled_price_tag>**: Set this to the price tag name for overriding a scaled amount.

The following sample XML entries add the **PRICE1** pricing tag to fixed balance impacts, and the **PRICE2** pricing tag to scaled balance impacts:

```
<?xml version="1.0" encoding="UTF-8"?>
<price_list xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="price_list.xsd" version="7.2">
<rate type="default" prorate_first="prorate" prorate_last="full"
step_type="total_quantity_rated">
   <description>$9.95 a month</description>
   <step_resource_id>840</step_resource_id>
   <quantity_tier>
      <balance_impact scaled_unit="none" flag="sponsorable discountable
proratable">
         <resource_id>840</resource_id>
         <glid>102</glid>
         <fixed_amount>10.0</fixed_amount>
         <fixed_price_tag>PRICE1</fixed_price_tag>
         <scaled_amount>5.0</scaled_amount>
         <scaled_price_tag>PRICE2</scaled_price_tag>
      </balance_impact>
   </quantity_tier>
</rate>
</price_list>
```

In this example, BRM would charge:

- A default fixed amount of $10. When the **PRICE1** pricing tag matches a pricing tag and date range combination in the **/offering_override_values** object, BRM would charge the fixed amount defined in **/offering_override_values**.
- A default scaled amount of 5%. When the **PRICE2** pricing tag matches a pricing tag and date range combination in the **/offering_override_values** object, BRM would charge the scaled amount defined in **/offering_override_values**.

Allowing Customers to Exceed Their Credit Limit

When configuring rates in XML, you can specify what happens to subscriptions when customers exceed their credit limits.

To do so, set **type** in the **<rate>** element to one of the following:

- **normal**: Enforce the credit limit. The subscription succeeds, but customers stay within their credit limit. Prorate the subscription according to the available balance.
- **default**: The subscription succeeds, and all available balance is used. The remaining amount is tracked as a negative account balance to be paid at the next top up for prepaid customers or current bill for postpaid customers.
- **loan**: The subscription succeeds. If the customer is eligible for a loan, all available balance is used and a loan is granted for the remaining amount. If the customer is not eligible for a loan, the subscription fails.
  See "About Loans" in *BRM Configuring and Collecting Payments* for more information about loans.
- **deduct_rental**: The subscription succeeds, and all available balance is used. The remaining amount is recorded as an outstanding amount to be paid at the next top up for prepaid customers or the current bill for postpaid customers.
- **outstanding_amount**: The subscription succeeds without using the available balance. The entire amount is recorded as an outstanding amount to be paid at the next top up for prepaid customers or the current bill for postpaid customers.
- **rental_failure**: Enforce the credit limit. The subscription fails. After retrying the subscription the configured maximum number of times, a notification event is sent to an external system for further processing.
- **auto_renew_cancel**: Enforce the credit limit. The subscription fails. After retrying the subscription the configured maximum number of times, a notification event is sent to an external system for further processing, and the subscription is canceled. It will not automatically renew the next cycle.
- **skip_cycle**: The subscription succeeds without using the available balance. Services will still be available, but billing will be skipped for this cycle. During rating, all balance impacts are dropped, and the unrated quantity is set to 0. The cycle forward date is moved to the next cycle.
- **minimum_amount**: The subscription succeeds if the customer's balance is at least a configured minimum amount. If they have enough balance for the full amount, the full amount is charged. If they have less than the full amount, but more than the minimum amount, the minimum amount is charged. If they have less than the minimum amount, the subscription fails, and the behavior is the same as for **rental_failure**.
  For this option, you can also configure the following:
  - The minimum amount in the **<minimum_scaled_amount>** element. By default, this is set to **0**, which means the subscription fails if the balance is less than the total amount.

  - Whether or not to prorate the service according to the minimum amount in the **prorate_quantity** flag of the **<rate>** element. By default, this is set to **yes**, and the service will be prorated. Set this to **no** to grant the full service for the minimum amount.

  This option is helpful if you want to offer your customers a small amount of their service, such as a few MB of data or minutes as an emergency stop-gap until they have enough balance to afford the full service.

For the **rental_failure**, **auto_renew_cancel**, and **minimum_amount** options, you can also configure the following when defining the product:

- The maximum number of times to retry the subscription before it fails. Configure this in the **<max_retry_count>** element of the product. The default is **0**, which sends the failure notification immediately without retrying.
- The frequency of subsequent retries. Configure this in the charge offer's **<retry_unit>** and **<retry_offset>** elements. The values for **<retry_unit>** are:
  - **1**: Seconds
  - **2**: Minutes
  - **3**: Hours
  - **4**: Days
  - **5**: Months
  - **6**: Years
  - **10**: Weeks

  The default for **<retry_unit>** is 4 and **<retry_offset>** is 1.

The **/purchased_product** object stores the current number of retries for the subscription in the PIN_FLD_RETRY_COUNT field and the date and time of the next retry in the PIN_FLD_NEXT_RETRY_T field.

If you do not configure the retry parameters at the offer level, the value for the **max_retry_count** business parameter is used. By default, this is set to **0**, which fails the subscription immediately without retrying. See "Configuring the Maximum Number of Subscription Retries" in *PDC Creating Product Offerings* for more information about setting this parameter.

The following XML rate and product example specifies to charge a minimum amount and prorate the service when customers exceed their credit limit. It also specifies to retry failed subscriptions a maximum of 3 times, every 6 hours.

```
<rate prorate_first="prorate" prorate_last="prorate" prorate_quantity="yes"
step_type="total_quantity_rated" type="minimum_amount">
  <description>Monthly Cycle Forward $12</description>
  <step_resource_id>0</step_resource_id>
  <quantity_tier>
    <balance_impact flag="proratable" scaled_unit="none">
      <resource_id>840</resource_id>
      <glid>0</glid>
      <fixed_amount>2.0</fixed_amount>
      <scaled_amount>10.0</scaled_amount>
      <minimum_scaled_amount>5.0</minimum_scaled_amount>
    </balance_impact>
  </quantity_tier>
</rate>
...
```

```
<product date_range_type="EVENT_DATE" partial="no" type="subscription">
    <product_name>MCF - $30</product_name>
    <product_code>MCF30</product_code>
    <priority>0.0</priority>
    <description>Monthly cycle forward charges.</description>
    <permitted>/service/ip</permitted>
    <max_retry_count>3</max_retry_count>
    <retry_offset>6</retry_offset>
    <retry_unit>3</retry_unit>
</product>
```

You can also use the **importexportpricing** utility or the PCM_OP_PRICE_SET_PRICE_LIST opcode to update these values.

Transitioning Plans and Deals

When configuring plan or deal transitions in XML, you can specify how to apply charges when customers perform transitions in the middle of their billing cycle. To do so, set **proration_flag** under the **<transition>** element to one of the following:

- **force_prorate**: Prorates charges for both plans or both deals. This is the default.
- **chg_full_from_svc**: Applies the full charges for the original plan or deal, and applies no charges for the new deal or plan.
- **chg_full_to_svc**: Refunds the charges for the original plan or deal, and applies the full charges for the new plan or deal.

The following sample XML configures BRM to prorate charges from both deals when your customers transition from one deal to another in the middle of their billing cycle:

```
<transition fee_flag="3" service_type="/service/ip" type="1"
proration_flag="force_prorate">
   <from type="/deal">Deal A</from>
   <to type="/deal">deal_product_not_aligned_inadvance</to>
</transition>
```

Configuring Add-On Products in Deals

When creating deals in XML, you can add base products and add-on products. All products in deals are base products by default, which means they can be purchased without any prerequisites. Add-on products require customers to own a valid base product.

You customize a product in a deal to be an add-on product by including the **<validity_align_mode>** element under **<deal_product>**. The add-on product's validity start date is the end date of a product that you specify. For example, assume product A has a validity period from June 1 through June 15. If you specify to align add-on product B's validity period with product A, product B's validity start date would be June 15.

You specify the product on which to align an add-on product's validity dates by setting the **<validity_align_mode>** element to one of the following:

- **base**: Aligns the validity dates with the specified base product. If set to **base**, you must also include these elements:
  - **<base_product_name>** set to the name of the base charge offer on which to set validity dates.

  - **<base_product_code>** set to the code of the base charge offer on which to set validity dates.

- **any_base_earliest**: Aligns the validity dates with the active base charge offer that expires first.
- **any_base_latest**: Aligns the validity dates with the active base charge offer that expires last.
- **any_earliest**: Aligns the validity dates with the active charge offer that expires first.
- **any_latest**: Aligns the validity dates with the active charge offer that expires last.

The following sample XML entries create a deal that includes an add-on product with a validity date aligned with that of **BASE_PRODUCT_SMS**:

```
<deal customization_flag="optional" ondemand_billing="no">
   <deal_name>Dealp1d1</deal_name>
   <description>Dealp1d1</description>
   <permitted>/service/ip</permitted>
   <deal_product status="active">
      <product_name>p1</product_name>
      <product_code>p1</product_code>
      <quantity>1.0</quantity>
      <purchase_start unit="month">5</purchase_start>
      <purchase_discount>0.0</purchase_discount>
      <cycle_start unit="month">5</cycle_start>
      <cycle_discount>0.0</cycle_discount>
      <usage_start unit="month">5</usage_start>
      <usage_discount>0.0</usage_discount>
      <purchase_mode>0</purchase_mode>
      <validity_align_mode>base</validity_align_mode>
      <base_product_name>BASE_PRODUCT_SMS</
base_product_name>
      <base_product_code>abcd-kfhg-sdjd-fgjk</base_product_code>
   </deal_product>
</deal>
```

You can also create or modify a deal with add-on products by using the PCM_OP_PRICE_SET_PRICE_LIST opcode. However, after customers subscribe to your deals, further updates to the product have no effect on any purchased products.

You can override the add-on product's validity dates for a specific customer by using the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL or PCM_OP_CUST_COMMIT_CUSTOMER opcode. See "Overriding Add-On Product Validity Dates in a Bundle" in *BRM Opcode Guide* for more information.

Purchasing the Same Product or Discount Multiple Times

When creating deals in XML, you can specify what happens if customers purchase the same product or discount more than once. The additional product or discount can be purchased as a new subscription, as a replacement of the existing subscription, or as an extension of the existing subscription if the additional product is purchased within a specified grace period.

Set the value for the **<purchase_mode>** element under the **<deal_product>** or **<deal_discount>** element to one of the following:

- **0**: The additional product or discount is purchased as a new, unrelated subscription. The balance impacts and validity periods of the old and new subscription are completely independent. This is the default option.
- **1**: The additional product or discount is purchased as an extension to the existing subscription if it is purchased within the specified grace period. The new resources are added to the existing balance group. The validity period for the resource is then set to either the old validity end date or the new validity end date, whichever is later.

  For example, on June 1, a customer purchases a deal that grants 3GB of data, is valid for 7 days, and has a grace period of 4 days. On June 3, after using 1GB, they purchase the deal again. The new 3GB balance is added to the remaining balance, for a total remaining balance of 5GB. BRM compares the validity periods of the old and new product and sets the validity end date for the total balance to the later of the two end dates, June 10.
- **2**: The additional product or discount is purchased as an extension to the existing subscription if it is purchased within the specified grace period. The new resources are added to the existing balance group. The validity period for the resource is then set by adding the new validity period to whatever remains of the old validity period.

  For example, on June 1, a customer purchases a deal that grants 3GB of data, is valid for 7 days, and has a grace period of 4 days. On June 3, after using 1GB, they purchase the deal again. The new 3GB balance is added to the remaining balance, for a total of 5GB. BRM adds the validity period of the new product to the old deal, and sets the validity end date for the total balance to June 15.
- **3**: The additional product or discount replaces the existing subscription. The existing subscription is canceled, any configured proration is applied, and the new subscription is created, with all resources and validity periods set as though for a new purchase.

  For example, on June 1, a customer purchases a deal that grants 3GB of data and is valid for 7 days. On June 3, after using 1GB, they purchase the deal again. The old subscription is canceled, any remaining balance is prorated, and the new 3GB balance is created, with a validity end date of June 10.
- **4**: The additional product or discount is purchased as an extension to the existing subscription if it is purchased within the specified grace period. The new resources are added to a new balance group. The new balance's validity period starts on the purchase date, and ends either on the old validity end date or the new validity end date, whichever is later.

  For example, on June 1, a customer purchases a deal that grants 3GB of data, is valid for 7 days, and has a grace period of 4 days. On June 3, after using 1GB, they purchase the deal again. The new 3GB balance is added to a new balance group, separate from the remaining 2GB in the old balance group.

  The new balance is valid starting from June 3. BRM calculates the validity end date by comparing the validity periods of the old and new balances and setting the validity end date for the new balance to the later of the two end dates, June 10. The old balance retains the original validity dates.
- **5**: The additional product or discount is purchased as an extension to the existing subscription if it is purchased within the specified grace period. The new resources are added to a new balance group. The new balance's validity period starts on the end date for the old balance's validity and the end is set by adding any remaining validity from the old balance to the total validity of the new balance.

  For example, on June 1, a customer purchases a deal that grants 3GB of data, is valid for 7 days, and has a grace period of 4 days. On June 3, after using 1GB, they purchase the deal again. The new 3GB balance is added to a new balance group, separate from the remaining 2GB in the old balance group.

The new balance is valid starting June 8. BRM calculates the new balance's validity end date by adding the validity period of the new balance to what remains of the old balance, and setting the validity end date for the new balance to June 15. The old balance retains the original validity dates.

When you extend the existing product or discount by setting **<purchase_mode>** to **1**. **2**, **4**, or **5**, you can also include a grace period during which the additional product or discount must be purchased to qualify for the extension. To do so, set the following elements:

- **<grace_period_offset>**: Set this to the amount of time in the grace period. The default is **0**, which specifies a grace period that never ends.
- **<grace_period_unit>**: Set this to the unit of the grace period: **0** (seconds), **1** (minutes), **2** (hours), or **3** (days). The default is **2** (days).

The following shows a sample of the XML for a product and a discount in a deal with **purchase_mode** set. The sample product specifies to extend the existing subscription and add the new and existing validity periods. The product is extended only if the additional product is purchased within two days of the existing product's expiration date.

The sample discount specifies to replace the existing discount with the new discount.

```
<deal customization_flag="optional" ondemand_billing="no">
          <deal_name>IP Deal</deal_name>
          <description>IP deal with purchase mode</description>
          <permitted>/service/ip</permitted>
          <deal_product status="active">
                    <product_name>Extending Product</product_name>
                    <product_code>ExtendingProduct</product_code>
                    <quantity>1.0</quantity>
                    <purchase_start unit="month">5</purchase_start>
                    <purchase_discount>0.0</purchase_discount>
                    <cycle_start unit="month">5</cycle_start>
                    <cycle_discount>0.0</cycle_discount>
                    <usage_start unit="month">5</usage_start>
                    <usage_discount>0.0</usage_discount>
                    <purchase_mode>2</purchase_mode>
                    <grace_period_offset>2</grace_period_offset>
                    <grace_period_unit>3</grace_period_unit>
                    <renewal_mode>1</renewal_mode>
          </deal_product>
          <deal_discount status="active">
                    <discount_name>Overwriting Discount</discount_name>
              <discount_code>OverwritingDiscount</discount_code>
              <quantity>1.0</quantity>
              <purchase_mode>3</purchase_mode>
          </deal_discount>
</deal>
```

Aligning Recurring Charges and Product Validity to a Specific Day of the Month

When creating products in XML, you can use the **calendar_dom** and **cycle_fee_flags** fields to align recurring charges and product validity to a specific day of the month instead of the customer's billing date.

You can determine when to align recurring charges and product validity as follows:

- To align recurring charges and product validity to a specific day of the month, set **calendar_dom** to an integer from 1-31 and set **cycle_fee_flags** to **1**.
- To align recurring charges and product validity to the billing date, set **calendar_dom** to **0**. The value of **cycle_fee_flags** is irrelevant in this scenario.
- To align recurring charges to the billing date but align product validity to a specific day of the month, set **calendar_dom** to an integer from 1-31 and set **cycle_fee_flags** to **0**.

> **Caution:**
>
> You cannot change the **calendar_dom** value for a product after it has been purchased. If you create a new version of the product with a different value in **calendar_dom**, BRM does not update the value in any purchased products and continues to use the value that was configured for the initial purchase.

For example, to align cycle rates and validity for a product to the 22nd day of the month, the XML input file would include the following:

```
<product type="subscription" partial="no"
date_range_impact_type="INSTANTIATED_DATE">
    <product_name>Product for a Specific Day of the Month</product_name>
    <product_code>5fac608f-ba2b-4e83-adc3-7563e349158f</product_code>
    <priority>1</priority>
    <permitted>/service/telco/gsm/telephony</permitted>
    <event_rating_map tod_mode="start_time" timezone_mode="event"
min_unit="none" incr_unit="none" rounding_rule="nearest"
flag="not_rate_cancelled">
        <event_type>/event/billing/product/fee/cycle/cycle_forward_monthly</
event_type>
        <rum_name>Occurrence</rum_name>
        <min_quantity>0</min_quantity>
        <incr_quantity>1</incr_quantity>
        <rate_plan_name>Rate Plan for a Specific Day of the Month</
rate_plan_name>
        <rate_plan_code>Occurrence-1bb17036-84ef-48be-9458-3b2922cf5276</
rate_plan_code>
        <rate_plan tax_when="never">
            <rate_plan_name>Rate Plan for a Specific Day of the Month</
rate_plan_name>
            <rate_plan_code>Occurrence-1bb17036-84ef-48be-9458-3b2922cf5276</
rate_plan_code>
            <currency_id>840</currency_id>
            <event_type>/event/billing/product/fee/cycle/
cycle_forward_monthly</event_type>
            <advance_billing_offset>0</advance_billing_offset>
            <cycle_fee_flags>1</cycle_fee_flags>
            <rate_tier date_range_type="absolute">
                <rate_tier_name>RateTier_0</rate_tier_name>
                <priority>100</priority>
                <date_range relative_start_unit="none"
relative_end_unit="none">
```

```
                    ...

                </date_range>
            </rate_tier>
            <rate_tier date_range_type="absolute">
                <rate_tier_name>RateTier_1</rate_tier_name>
                <priority>90</priority>
                <date_range relative_start_unit="none"
relative_end_unit="none">

                    ...

                </date_range>
            </rate_tier>
        </rate_plan>
    </event_rating_map>
    <calendar_dom>22</calendar_dom>
</product>
```

If a customer with a billing date of the 14th of the month purchased the product in this example on January 18, their first cycle forward fee would be applied on January 18, prorated to cover the period between January 18 and January 22 (the **calendar_dom** date.) The next fee would be applied on January 22, at the full amount to cover the period between January 22 and February 22. Bills would be generated on the 14th of each month.

However, if the value of **cycle_fee_flags** in the example above was **0**, the first cycle forward fee would be applied on January 18, prorated to cover the period between January 18 and February 14 (the billing date). The next fee would be applied on February 14, at the full amount to cover the period between February 14 and March 14. In the final month of the product's duration, it would only be valid until the 22nd of the month (the **calendar_dom** date.) If the product was valid for three months, the final fee would be prorated to cover the period from March 14 to March 22.

> **Note:**
>
> For products in deals that are suspended and then reactivated, you can specify that the cycle aligns with the reactivation date rather than the original purchase date using the **renewal_mode** element under the **deal_product** element. See "Setting Product Cycle Alignment for Reactivated Deals" for information about how **renewal_mode** interacts with **calendar_dom** and **cycle_fee_flags**.

Splitting Noncurrency Balances into Multiple Validity Periods

When creating products in XML, you can grant noncurrency balances in smaller portions on an incremental basis. For example, you could grant 30 Gigabytes with a one-month validity that is distributed as 1 Gigabyte per day.

Set the **<split_bucket>** element under the **<balance_impact>** element as follows:

<split_bucket validity="*validityType***" unit="***unit***">***amount***</split_bucket>**

where:

- *validityType* is either **bucket** (the balance expires at the end of the incremental validity period) or **total** (the balance expires at the end of the total validity period).
- *amount* and *unit* specify the length of each incremental validity period, such as five hours or two days.

For example, to create an incremental validity period of four hours that has its noncurrency resources expire at the end of the total validity period:

```
<product>
   <event_rating_map>
      <rate_plan>
         <rate_tier>
            <rate>
               <quantity_tier>
                  <balance_impact scaled_unit="none" flag="discountable
grantable sponsorable">
                     <resource_id>100002</
resource_id>
                     <impact_category>default</
impact_category>
                     <glid>0</glid>
                     <fixed_amount>0.0</
fixed_amount>
                     <scaled_amount>-233.0</
scaled_amount>
                     <split_bucket validity="total" unit="hour">4</
split_bucket>
                  </balance_impact>
               </quantity_tier>
            </rate>
         </rate_tier>
      </rate_plan>
   </event_rating_map>
</product>
```

Setting Product Cycle Alignment for Reactivated Deals

When creating deals in XML, you can specify when the cycle should align for products if a customer suspends and then reactivates the deal.

Set the value for the **renewal_mode** element under the **deal_product** element to **1** or **0**. This setting is not valid for discounts.

If **renewal_mode** is set to **0**, then the cycle aligns as described in "Aligning Recurring Charges and Product Validity to a Specific Day of the Month" for more information.

If **renewal_mode** is set to **1**, the following scenarios are possible:

- When **cycle_fee_flags** under **rate_plan** is set to **1**:
  - If **calendar_dom** under **product** is not set, then the cycle aligns with the reactivation date.
  - If **calendar_dom** is set, the cycle aligns with either **calendar_dom** or the reactivation date, whichever is later.
- When **cycle_fee_flags** is set to **0**, the cycle aligns with the billing date.

For a sample of the XML for a product in a deal with **renewal_mode** set to **1**, see "Purchasing the Same Product or Discount Multiple Times" for more information.

You can update the value of **renewal_mode** by using the PCM_OP_PRICE_SET_PRICE_LIST opcode, but after customers subscribe to your deals, further updates to **renewal_mode** have no effect on the purchased products.

Activating Products in Plans and Deals on First Usage

When creating plans or deals in XML, you can specify to activate all products in a plan or deal when one of its products is activated on first usage. To do so, you use the **first_usage** flag in the **<deal>** element (for deals) and in the **<plan>** element (for plans).

Set the **first_usage** flag to one of the following:

- **yes**: Activates all products in a deal or plan upon the first usage of any of its products.

  For example, assume a deal contains products A, B, and C. If the first service a customer uses is in product B, all three products (A, B, and C) are activated at that time.

- **no**: When one product in a deal or plan is activated on first usage, the other products in the plan or deal are not activated.

  For example, assume a plan contains products A, B, and C. If the first service a customer uses is in product B, only product B is activated at that time. This is the default.

The following shows an example of a plan with the **first_usage** flag set:

```
<plan ondemand_billing="no" first_usage="yes">
   <plan_name>Plan01</plan_name>
   <description>Plan01</description>
   <service_deal>
      <service_name>/service/ip</service_name>
      <deal_name>Deal01</deal_name>
      <bal_info_index>0</bal_info_index>
      <subscription_index>0</subscription_index>
   </service_deal>
</plan>
```

Using Date Ranges for Versioning

When creating products in XML, you can add rate plans with new date ranges to create new versions of the same product.

You use the **date_range_type** attribute for the product to specify whether existing subscriptions move to the new charge or continue with the old charge.

BRM determines which rate plan to apply based on the value of **date_range_type** as follows:

- **EVENT_DATE**: The charge event date determines which rate plan is applied. Any existing subscriptions will use the new rate plan after the existing subscription's end date. This is the default behavior.
- **PURCHASE_DATE**: The original purchase date determines which rate plan is applied. Any existing subscriptions will continue to use the old rate plan. Any new subscriptions purchased within the new rate plan's date range will use the new rate plan.

> **Note:**
>
> If a customer transfers their subscription as part of a service group transfer and **date_range_type** is set to **PURCHASE_DATE**, the rate plan will be determined based on the service transfer date, not the original purchase date.

- **INSTANTIATED_DATE**: The original service instantiation date determines which rate plan is applied. Any existing subscriptions will continue to use the old rate plan. Any new subscriptions instantiated during the new rate plan's date range will use the new rate plan. This option is useful if your services often have a long delay between purchase and instantiation date. It allows customers with existing subscriptions to continue with the old rate plan, while those who purchase under the old rate plan can take advantage of the new rate plan if it comes into effect before their new service is available.

The following shows an example of a product using **date_range_type**:

```
<?xml version="1.0" encoding="UTF-8"?>
<price_list version="7.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="price_list.xsd">
    <product partial="no" type="subscription" date_range_type="EVENT_DATE">
        <product_name>prod1</product_name>
        <product_code>prod1</product_code>
            ...
    </product>
</price_list>
```

The following examples describe how the value of **date_range_type** affects the rate plan applied. In the examples, you have an existing product with a rate plan that credits 1GB of free data every week. You add a new rate plan crediting 1.5GB, set the end date of the old rate plan to June 10, and set the start date for the new rate plan to June 11. Depending on the value of **date_range_type** for the product, the following happens:

- If **dateRangeImpactType** is set to **EVENT_DATE**, all subscriptions get 1GB until June 10 and 1.5GB thereafter.
- If **dateRangeImpactType** is set to **PURCHASE_DATE**, subscriptions purchased before June 10 continue to get 1GB, even for events occurring after June 10. Subscriptions purchased after June 10 get 1.5GB.
- If **dateRangeImpactType** is set to **INSTANTIATED_DATE**, subscriptions instantiated before June 10 continue to get 1GB, even for events occurring after June 10. Subscriptions instantiated after June 10 get 1.5GB, even if they were purchased before.

Defining Product Specification Attributes for Pricing Components

When creating products, discounts, chargeshares, deals, and packages in XML, you can define product specification attributes to store information used by external applications, such as enterprise product catalogs, beyond what is provided in BRM.

You define product specification attributes in **attributes** elements with nested **name** and **value** elements. The following examples show product specification attributes for each pricing component:

- Products:

```
<product partial="no" type="subscription">
    <product_name>Product with product specification attributes</
product_name>
    <product_code>Product 1</product_code>
    <priority>0.0</priority>
    <description>cycle_forward_monthly</description>
    <permitted>/service/ip</permitted>
    ...
    <attributes>
        <name>Lifecycle Status</name>
        <value>Test</value>
    </attributes>
    <attributes>
        <name>External Reference ID</name>
        <value>09876</value>
    </attributes>
    ...
</product>
```

- Discounts:

```
<discount mode="parallel" type="subscription">
        <discount_name>Discount with product specification attributes</
discount_name>
        <discount_code>Discount 1</discount_code>
        <priority>0.0</priority>
        <permitted>/service/ip</permitted>
        ...
    <attributes>
                <name>Lifecycle Status</name>
                <value>Suspend</value>
        </attributes>
        <attributes>
                <name>External Reference ID</name>
                <value>98645</value>
    </attributes>
    ...
</discount>
```

- Chargeshares:

```
<sponsorship>
        <sponsorship_name>Standard GSM Charge Share</sponsorship_name>
    ...
        <attributes>
                <name>Lifecycle Status</name>
                <value>Retired</value>
        </attributes>
        <attributes>
                <name>External Reference ID</name>
                <value>24601</value>
        </attributes>
```

```
    ...
</sponsorship>
```

- Deals:

```
<deal customization_flag="optional" ondemand_billing="no">
    <deal_name>Deal with product specification attributes</deal_name>
        <permitted>/service/ip</permitted>
        <deal_product status="active">
            <product_name>Product with product specification attributes</
product_name>
            <product_code>Product 1</product_code>
            <quantity>1.0</quantity>
        </deal_product>
        <deal_discount status="active">
            <discount_name>Discount with product specification attributes</
discount_name>
            <discount_code>Discount 1</discount_code>
            <quantity>1.0</quantity>
        </deal_discount>
    ...
        <attributes>
                <name>Lifecycle Status</name>
                <value>Draft</value>
        </attributes>
        <attributes>
                <name>External Reference ID</name>
                <value>8675309</value>
    </attributes>
    ...
</deal>
```

- Plans:

```
<plan ondemand_billing="no">
        <plan_name>Plan with product specification attributes</plan_name>
        <service_deal>
            <service_name>/service/ip</service_name>
            <deal_name>Deal with product specification attributes</
deal_name>
        </service_deal>
    ...
        <attributes>
                <name>Lifecycle Status</name>
                <value>Release</value>
        </attributes>
        <attributes>
                <name>External Reference ID</name>
                <value>011235813</value>
        </attributes>
    ...
</plan>
```

Although BRM does not provide validation for these attributes by default, you can configure validations in the PCM_OP_SUBSCRIPTION_POL_VALIDATE_OFFERING opcode. See "Validating Extended Attributes" in *BRM Opcode Guide*.

Storing and Managing Product Specification Attributes

When you import your pricing objects, product specification attributes are stored in an **/offer_attribute_group** object. The POID for the **/offer_attribute_group** object is stored in the **PIN_FLD_ATTRIBUTE_OBJ** field of the pricing object. For example, the flist for a deal with product specification attributes would contain the following:

```
0    PIN_FLD_POID    POID [0] 0.0.0.1 /deal 95359 0
0    PIN_FLD_ATTRIBUTE_OBJ    POID [0] 0.0.0.1 /offer_attribute_group 96383 0
```

You can use the loadpricelist utility to manage product specification attributes as follows:

- Load a price list to add or modify product specification attributes. The values in the **/offer_attribute_group** objects are updated, but the **PIN_FLD_ATTRIBUTE_OBJ** value for the pricing object remains the same.
- Load a price list to remove product specification attributes from a pricing object. The attributes are removed from the **/offer_attribute_group** object. If you remove all product specification attributes from a pricing object, the value of **PIN_FLD_ATTRIBUTE_OBJ** is changed to a null value (**POID [0] 0.0.0.0 0 0**), and the empty **/offer_attribute_group** object remains.
- Download a price list that includes product specification attributes from the database. The utility converts PIN_FLD_ATTRIBUTE_OBJ to **attributes** elements with the appropriate **name** and **value** elements in the output XML.
- Delete a pricing object from the database. Any associated **/offer_attribute_group** objects are also deleted.

Stop Rating Inactive, Canceled, or SuspendedActive Accounts

When configuring cycle rates and purchase rates in XML, you can specify whether BRM should continue charging or stop charging accounts that have the following:

- An **Inactive** status
- A **Cancelled** status
- A **SuspendedActive** custom life cycle state

> **Note:**
>
> This option is valid only if your system includes a custom life cycle state named **SuspendedActive**. See "Creating Custom Service Life Cycles" in *BRM Managing Customers*.

For each cycle and purchase rate defined in your pricing XML file, set the **<event_rating_map> flag** element to one of the following:

- **not_rate_inactive**: Specifies not to apply charges to inactive accounts.
- **not_rate_cancelled**: Specifies not to apply charges to canceled accounts.
- **rate_lifecycle_suspend**: Specifies to continue applying charges to suspended accounts.

For example, the following specifies to continue applying charges to suspended accounts:

```
<event_rating_map flag="rate_lifecycle_suspend" incr_unit="none"
min_unit="none" rounding_rule="nearest" timezone_mode="event"
tod_mode="start_time">
```

Setting Loan Thresholds for Plans

When creating credit limits for plans in XML, you can use the following elements to set balance thresholds below which a customer is offered a loan:

- **<thresholds_loan>**: When this percent of the balance is consumed, the customer is offered a loan. The percentage must be in 5% increments.
- **<thresholds_fixed_loan>**: When the balance falls below this fixed number, which must be greater than the credit floor and less than the credit ceiling, the customer is offered a loan.

You can set multiple thresholds within the elements, delimited by a space.

When the customer subscribes to the plan, the thresholds are stored in the **/config/credit_profile** object associated with their balance group. For more information about loan thresholds and what happens when they are breached, see "About Loan Thresholds" in *BRM Configuring and Collecting Payments*.

The following sample XML entries specify several threshold elements. The **<thresholds_loan>** and **<thresholds_fixed_loan>** elements are used for offering loans and the **<thresholds>** and **<thresholds_fixed>** elements are used for different notifications or automatic top-ups:

```
<plan ondemand_billing="no">
      <plan_name>Plan for Loan Thresholds</plan_name>
      <plan_code>123456</plan_code>
      <description />
      <credit_limit>
         <resource_id>840</resource_id>
         <limit>100</limit>
         <thresholds>30 60 90</thresholds>
         <thresholds_fixed>25 50 75</thresholds_fixed>
         <thresholds_loan>35 65 95</thresholds_loan>
         <thresholds_fixed_loan>20 60 80</thresholds_fixed_loan>
         <floor>10</floor>
      </credit_limit>
      ...
   </plan>
```

Enabling Dynamic Credit Floors in Plans

You can set BRM to generate a dynamic credit floor from the granted amounts from the sub-balances that are valid for the current cycle for the resource (for example, minutes). To do so, you use the **dynamic_floor** flag in the **<credit_limit>** element.

Set the **dynamic_floor** flag to one of the following:

- **1**: Sets the credit floor to be based on the granted amounts from the sub-balances that are valid for the current cycle for the resource (for example, minutes).

For example, assume a customer has a usual monthly grant of 10GB of data, but for a single charge period has purchased a booster pack of 5 GB of data. The dynamic credit floor updates the credit floor to 15 GB for the current cycle, and then reverts to 10 GB at the start of the next cycle.

- **0**: Does not set a dynamic credit floor. In this case, the fixed credit floor will be used. This is the default.

The following shows an example of a plan with the **dynamic_floor** flag set:

```
<plan ondemand_billing="no">
    <plan_name>Plan for Loan Thresholds</plan_name>
    <plan_code>123456</plan_code>
    <description />
    <credit_limit>
        <resource_id>840</resource_id>
        <limit>100.0</limit>
        <thresholds>50 60</thresholds>
        <thresholds_fixed>80</thresholds_fixed>
        <thresholds_loan>20 40</thresholds_loan>
        <thresholds_fixed_loan>40</thresholds_fixed_loan>
        <floor>0.0</floor>
        <dynamic_floor>1</dynamic_floor>
    </credit_limit>
    ...
 </plan>
```

Creating Tax Selectors for Products

You can create tax selectors for products so that defined tax codes can be applied based on attributes in the **/account**, **/service**, **/event**, or **/profile** objects.

To do so, you define the selector using the **<tax_selector>** element and then set the name of the selector in the **<tax_selector_name>** element under the **<product>** element.

The elements under the **<tax_selector>** element are:

- **selector_name**: Text containing the name of the selector.
- **selector_rules**: Container for a set of selection criteria.
- **criteria**: The selection criteria for the tax code. This has an attribute, **operator**, which specifies the comparison type. It must be either **equal** or **in**. The operator is case-sensitive, and the default is **equal**.
- **object_class**: The name of the class containing the element to be compared.
- **elem_name**: The specific child element of the object in the **<object_class>** element that contains the field to be compared. This can contain multiple layers, separated by periods, for example, **SUB_BAL_IMPACTS.SUB_BALANCES**.
- **field_name**: The name of the field containing the data to be compared.
- **field_value**: The value that is to be compared with the contents of the field specified by **<field_name>**.
- **code**: The tax code that should be mapped if the comparison is successful.

The following shows an example of **<tax_selector>** configuration:

```
<tax_selector>
  <selector_name>tax1</selector_name>
  <selector_rules>
```

```
    <criteria operator="equal">
      <object_class>/account</object_class>
      <elem_name> PIN_FLD_NAMEINFO </elem_name>
      <field_name>PIN_FLD_CITY</field_name>
      <field_value>Bangalore</field_value>
    </criteria>
    <criteria operator="equal">
      <object_class>/event</object_class>
      <field_name>PIN_FLD_NAME</field_name>
      <field_value>cycle_fwd</field_value>
    </criteria>
    <code>VAT</code>
  </selector_rules>
  <selector_rules>
    <criteria operator="equal">
      <object_class>/account</object_class>
      <elem_name> PIN_FLD_NAMEINFO </elem_name>
      <field_name>PIN_FLD_CITY</field_name>
      <field_value>VSKP</field_value>
    </criteria>
    <criteria operator="equal">
       <object_class>/event</object_class>
       <field_name>PIN_FLD_NAME</field_name>
       <field_value>cycle_arrear</field_value>
    </criteria>
    <tax_code>VAT1</tax_code>
  </selector_rules>
</tax_selector>
```

Following is an example of a product configuration containing the **<tax_selector_name>** element:

```
<product date_range_type="EVENT_DATE" partial="no" type="subscription">
. . .
  <event_rating_map incr_unit="none" min_unit="none" rounding_rule="nearest"
timezone_mode="event" tod_mode="start_time">
. . .
    <rate_plan advance_billing_unit="day" tax_when="now">
      <rate_plan_name>charge2$-session</rate_plan_name>
      <currency_id>840</currency_id>
      <event_type>/event/session</event_type>
      <tax_selector_name>tax1</tax_selector_name>
      <advance_billing_offset>0</advance_billing_offset>
      <cycle_fee_flags>0</cycle_fee_flags>
. . .
    </rate_plan>
. . .
  </event_rating_map>
</product>
```

Creating Tax Exemption Selectors for Products

You can create tax exemption selectors for products so that defined tax exemptions can be applied based on attributes in the **/account**, **/service**, **/event**, or **/profile** objects. For more information about tax exemptions, see "About Tax Exemptions" in *BRM Calculating Taxes*.

To do so, you define the selector using the **<tax_exemption_selector>** element and then set the name of the selector in the **<selector_name>** element under the **<product>** element.

The elements under the **<tax_exemption_selector>** element are:

- **selector_name**: Text containing the name of the selector.
- **selector_rules**: Container for a set of selection criteria.
- **criteria**: The selection criteria for the tax exemption. This has an attribute, **operator**, which specifies the comparison type. It must be either **equal** or **in**. The operator is case-sensitive, and the default is **equal**.
- **object_class**: The name of the class containing the element to be compared.
- **elem_name**: The specific child element of the object in the **<object_class>** element that contains the field to be compared. This can contain multiple layers, separated by periods, for example, **SUB_BAL_IMPACTS.SUB_BALANCES**.
- **field_name**: The name of the field containing the data to be compared.
- **field_value**: The value that is to be compared with the contents of the field specified by **<field_name>**.
- **code**: The exemption code that should be mapped if the comparison is successful.

The following shows an example of **<tax_exemption_selector>** configuration:

```
<tax_exemption_selector>
  <selector_name>tax_exempt</selector_name>
  <selector_rules>
    <criteria operator="equal">
      <object_class>/account</object_class>
      <elem_name> PIN_FLD_NAMEINFO </elem_name>
      <field_name>PIN_FLD_CITY</field_name>
      <field_value>Bangalore</field_value>
    </criteria>
    <criteria operator="equal">
      <object_class>/event</object_class>
      <field_name>PIN_FLD_NAME</field_name>
      <field_value>cycle_fwd</field_value>
    </criteria>
    <code>student</code>
  </selector_rules>
  <selector_rules>
    <criteria operator="equal">
      <object_class>/account</object_class>
      <elem_name> PIN_FLD_NAMEINFO </elem_name>
      <field_name>PIN_FLD_CITY</field_name>
      <field_value>VSKP</field_value>
    </criteria>
    <criteria operator="equal">
      <object_class>/event</object_class>
```

```
      <field_name>PIN_FLD_NAME</field_name>
      <field_value>cycle_arrear</field_value>
    </criteria>
    <code>employee</code>
  </selector_rules>
</tax_exemption_selector>
```

Following is an example of a product configuration containing the **<selector_name>** element:

```
<product partial="no" type="subscription">
  <product_name>tax_prod_RPS</product_name>
  <product_code>tax_prod_RPS</product_code>
  <priority>0.0</priority>
  <description>Sample product</description>
  <permitted>/service/ip</permitted>
  <tax_supplier>TS1</tax_supplier>
  <selector_name>tax_exempt</selector_name>
. . .
</product>
```

26

Real-Time Rating Based on Multiple RUMs

This chapter shows how to use Oracle Communications Billing and Revenue Management (BRM) to set up products to rate events based on multiple ratable usage metrics (RUMs) if you are using Pricing Center.

Topics in this document:

About Rating Based on Multiple RUMs

By default, BRM rates an event by using a single RUM that is specified in the product's usage map in Pricing Center. You can also set up your products to rate an event based on multiple RUMs. For example, you can set up a Session Event with both Occurrence and Duration selected as RUMs by which the event is measured.

To rate an event by using multiple RUMs, you map the event to the RUMs in the product's usage map.

To apply the RUMs, you map the event to the RUMs and the rate plan in the product's usage map. For example, you can map two IP Dialup events to the RUMs, **Duration** and **Occurrence**, in a usage map for an Internet Access product. You can select both the events for rating.

For each event to be rated, click the check box in the ***** column, located between the event number column and the **Event** column.

> **Note:**
>
> Specifying multiple RUMS for the same event type and selecting events to be rated are two separate functions, each handled in the Event Map.

When an event is rated using multiple RUMs, a balance impact is applied for each RUM per each resource. For example, if two resources are impacted, two separate balance impacts are created.

For information about setting up products with multiple RUMs, see "Applying Multiple RUMs to Rate an Event".

Applying Multiple RUMs to Rate an Event

To rate an event by using multiple RUMs, you map the event to the RUMs in the product's usage map.

To use multiple RUMs to rate an event, you do the following:

1. Define the RUMs available for rating. See "About Ratable Usage Metrics".
2. Map the event to the RUMs in the product's usage map. See "Example of Mapping an Event to Multiple RUMs".

Example of Mapping an Event to Multiple RUMs

When you create a product, you specify the events to rate for the service, and the rate plans that determine how to charge for those events. To rate an event by using multiple RUMs, you associate the event with the RUMs and the rate plans.

You can map the same event to multiple RUMs by using different rate plans. For example, you can specify a rate plan to charge for a fax event based on the number of pages faxed and another rate plan to charge based on the duration of the fax session.

To map an event to RUMs:

1. Specify the RUMs in the product's usage map.
2. Indicate the events to be rated by clicking the check box in the ***** column, located between the event number column and the event type column.
3. When you finish creating products, deals, and plans, commit your price list to the BRM database.

27

Rating Based on Multiple RUMs with Pipeline Manager

This chapter shows how to use Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager to set up charge offers or rate plans to rate batch events based on multiple ratable usage metrics (RUMs).

Topics in this document:

About Rating Based on Multiple RUMs with Pipeline Manager

To rate an event using multiple RUMs with Pipeline Manager, you set up multiple Pricing steps. In other words, you configure pipeline rating within the individual pricing.

For each RUM, the data is set in the Event Data Record (EDR) before it is sent to the batch pipeline. Pipeline Manager then uses this data to do the rating based on each individual RUM. Each RUM is fully rated before rating the next RUM.

For more information about applying multiple RUMs to rate a batch event, see "Applying Multiple RUMs to Rate a Batch Event".

Applying Multiple RUMs to Rate a Batch Event

To rate a batch event by using multiple RUMs, you set up multiple Pricing steps.

To use multiple RUMs to rate a batch event, do the following:

1. Define the RUMs available for rating.
2. Set up pricing steps. The following example shows a multi-RUM charge model created for a standard GPRS delayed event:
   - Duration RUM: 0.02 EUR/min
   - Volume Sending RUM: 2 EUR/MB
   - Volume Receiving RUM: 2 EUR/MB
   - In this case, you set up the following pricing steps:

     Configure the pricing for the **Duration RUM** as follows:

     Beat: 60

     Charge: 0.02

     Charge Base: 60

     Configure the pricing for the **Volume Receiving RUM** as follows:

     Beat: 2048

Charge: 2.00

Charge Base: 1048576

Configure the pricing for the **Volume Sending RUM** as follows:

Beat: 2048

Charge: 2.00

Charge Base: 1048576

28

Real-Time Rating Based on Date and Time

This chapter explains how to use Oracle Communications Billing and Revenue Management (BRM) to set up rates based on date and time for real-time rating.

Topics in this document:

About Rating Based on Date and Time

BRM rates billable events by capturing data about the events (for more information, see "About Creating a Price List"), such as the date and time each event occurred. Because BRM is aware of the date and time of an event, the rating opcodes can use the date and time to determine how to rate the event.

Ways to Specify When Rates Are Valid

You can specify when rates are valid in the following ways:

- **In products.** You can specify when a product is valid, including all the rates it contains. See "Restricting When a Product Is Available".
- **In rate tiers.** You can use rate tiers to specify a set of rates, each based on a specific time or day. See "Using Rate Tiers to Set Up Rating Based on Date and Time".

When you create rates based on date and time, you can specify how to recognize the time that the event occurred (for example, at the start, end, or duration of the event).

You specify a time-of-day mode to indicate when a rate should be applied relative to either an event's start time, its end time, or the difference between the two. This is especially useful when an event overlaps a time boundary where a rate changes.

For example, 12 a.m. Monday might be the boundary between a special weekend rate and a regular weekday rate:

- To apply the special weekend rate to the entire event, specify **Start Time**.
- To apply the regular weekday rate to the entire event, specify **End Time**.
- To apply the weekend rate to the portion of the event before midnight and the weekday rate after midnight, specify **Timed**.

> **Note:**
>
> Midnight (12 a.m.) is shown as 00.00.00.

Using Rate Tiers to Set Up Rating Based on Date and Time

You use rate tiers to set up valid dates and times for rates. You can include multiple rate tiers in a rate plan.

Specifying Which Rate Tiers Are Applied First

As you create rate tiers, they appear on a list. BRM applies rates from the top of the list down. You can change the order in which rate tiers are applied by changing their places in the list.

For example, at the top of the rate tier list you could insert a promotional rate tier valid only until the end of May 2003. BRM applies that rate only until May 31, 2003. After that, the promotional rate tier is no longer valid, and BRM no longer applies it.

Specifying How Long Rate Tiers Are Valid

You specify how long a rate tier is valid by specifying a duration. You can specify that a rate tier is always valid or is valid for either an absolute or a relative date range.

For example, your basic IP Fax rate tier should not change and always applies. You could specify an absolute date range of 12/1/01 through 12/31/01 for a promotional holiday rate tier. You could also specify a relative date range of 30 days from purchase until 60 days from purchase to give your customers special rates during the second month.

Specifying When Rate Tiers Are Valid

You specify when a rate tier is valid by selecting ranges of dates, days of the week, and times of day. For example:

- For each rate tier, you specify a valid date range. For example, you can specify that a rate tier is valid for the entire year 2001.
- For each date range, you specify one or more valid day-of-the-week ranges. For example, you can specify that a rate tier is valid Monday through Friday.
- For each day-of-the-week range, you specify one or more valid time-of-day ranges. For example, you can specify that a rate tier is valid between 6 a.m. and 8 p.m.

Using Date, Day, and Time Ranges Together

Date ranges consist of a start date and an end date between which a rate tier is valid. For example, a promotional rate tier for usage fees is valid only for the month of January.

Day ranges consist of a list of days during which a rate is valid. For example, a rate tier for off-peak IP usage fees is valid only on weekends.

Time-of-day ranges consist of a start time and an end time between which a rate tier is valid. For example, an after-hours rate tier for IP usage fees is valid only between 10 p.m. and 6 a.m.

Within each date range, you can use multiple day ranges; within each day range, you can use multiple time-of-day ranges.

Using the Correct Time Zone

Because BRM uses the date and time of an event to determine how to rate the event, using the correct time zone during rating is important. You specify whether real-time rating uses the *event* time zone or the *server* time zone to rate the event. By using time zone data, BRM can correctly calculate event start and end times before rating occurs.

Specifying a Time Zone Mode

When you create a product in Pricing Center, you specify which of the following time zone modes BRM uses for rating events whose rates are based on time of day:

- **Event:** BRM uses the time zone in which the event occurs. For example, if a user in California logs on to a terminal server located in California, BRM uses California time, even if the BRM server that rates the event is in New York. This is the default time zone mode.
- **Server:** BRM uses the time zone in which the BRM server that rates the event is located. For example, if a California user logs on to a terminal server located in California but the BRM server that rates the event is in New York, BRM uses New York time. For information about setting your system's server time zone, see "Resetting the Server Time Zone".

For information about setting the time zone mode, see the Pricing Center Help.

Resetting the Server Time Zone

The server time zone is where the server you use for rating resides.

By default, the server time zone is set to **America/Los_Angeles**. If your rating server is not in that time zone, reset the server time zone to the time zone ID indicating the server location.

> **Note:**
>
> BRM supports only the time zone IDs defined in instances of the Java TimeZone class.

To reset the server time zone:

1. Get a list of time zone IDs in the Java TimeZone class by running the following function:

   ```
   TimeZone.getAvailableIDs()
   ```

2. In the list, find the ID for the time zone in which your server resides and write it down.
3. Open the *BRM_home***/sys/cm/timezones.txt** file, where *BRM_home* is the directory in which BRM is installed.

   This file lists all the time zone IDs available for use in your system.
4. If the ID for the time zone in which your server resides is *not* in the **timezones.txt** file, do the following:
   a. Open the *BRM_home***/sys/cm/sample_timezone.txt** file.

b. Add an entry to the **sample_timezone.txt** file for the time zone in which your server resides by following the instructions in the file.

> **Note:**
>
> BRM supports only the time zone IDs defined in instances of the Java TimeZone class.
>
> Each time zone ID entry in the **sample_timezone.txt** file contains 13 mandatory parameters. The value of the **tzone_name** parameter must match the name of a time zone ID in the Java TimeZone class. To get a list of time zone IDs in the TimeZone class, run the `TimeZone.getAvailableIDs()` function.

> **Note:**
>
> To add multiple time zone IDs to the file, enter each time zone ID entry on a separate line.

c. Save and close the **sample_timezone.txt** file.

5. Open the *BRM_home*/**sys/cm/pin.conf** file.
6. If you added an entry to the **sample_timezone.txt** file in step 4, add the following entry to the **pin.conf** file:

   ```
   - fm_rate timezone_file sample_timezone.txt
   ```

7. Find the following entry in the **pin.conf** file:

   -rating_timezone *server_timezone_ID*

   where *server_timezone_ID* specifies the ID of the time zone in which the BRM server performing rating is located.

8. Replace the current *server_timezone_ID* value with the time zone ID of your BRM rating server.

   For example, if your rating server is in Saskatchewan, use the following time zone ID:

   ```
   - fm_rate rating_timezone Canada/Saskatchewan
   ```

   > **Note:**
   >
   > To add multiple time zones to the file, add separate **-fm_rate rating_timezone** entries.

9. Save and close the file.
10. Stop and restart the CM.

Configuring Applications to Generate Time Zone Data

When a BRM client application or service integration client (such as RADIUS Manager and Universal Event Loader) generates an event, that event includes time zone data. You must

configure event-generating applications to generate the correct time zone data. This means creating an entry in a **pin.conf** or properties file. See the configuration section of your client application or the service integration component documentation for details.

Using Event Start Time to Determine a Product's Validity

By default, in real-time rating an event's end time is used to determine a product's validity.

You can specify that the event start time be used for determining the product's validity by adding an entry to the CM configuration file.

To use the event start time to determine the product's validity:

1. Open the *BRM_Home***/sys/cm/pin.conf** file.
2. Add the following entry:

   ```
   - fm_rate use_prods_based_on_start_t /event
   ```

   You can use **/event** to use the start time for all events, or use a subclass of **/event** to specify just events of a given type.
3. Stop and restart the CM.

Dynamically Changing One-Time and Recurring Fees Based On the Date

You can configure BRM to dynamically change the rate for one-time and recurring fees per customer based on the date. For example, when a customer cancels a product, you could charge a default amount of $50, but charge $45 in June and $40 in July.

You set up BRM to dynamically change one-time and recurring fees by doing the following:

- Creating a pricing tag and specifying the event type, date ranges, and rates associated with it. For example, you could create a pricing tag named **Special2032** that is associated with the following:
  - Event type: **/event/billing/product/fee/cycle/cycle_forward_monthly**
  - Date range: January 1, 2032 through February 3, 2032
  - Rate: $100

  BRM stores the pricing tag data in **/offering_override_values** objects.
- Adding the pricing tag, such as **Special2032**, to the desired balance impacts in your price list.

When rating events, BRM determines whether the pricing tag in a balance impact matches a pricing tag and date range combination defined in the **/offering_override_values** object. If it does, BRM charges the rate associated with the pricing tag. If it does not find a match, BRM charges the default rate.

To dynamically change one-time and recurring fees, do the following:

1. Define each pricing tag and the event type, time frame, and rate associated with it. To do so, customize your third-party client application to call the Subscription opcodes. See "Overriding Charges and Discounts for a Period of Time" in *BRM Opcode Guide*.

> **Note:**
>
> Each pricing tag name must be unique, and the time frames cannot overlap.

2. Add pricing tags to your balance impacts in one of the following ways:
   - Using the **loadpricelist** utility. In your price list XML file, set the **<fixed_price_tag>** and **<scaled_price_tag>** elements under the **<balance_impact>** element. See "Adding Pricing Tags to Rates".
   - Using a third-party client application that calls the PCM_OP_PRICE_SET_PRICE_LIST opcode. In the opcode's input flist, set the PIN_FLD_FIXED_PRICED_TAG and PIN_FLD_SCALED_PRICE_TAG fields to the name of the appropriate pricing tag.

     For more information about PCM_OP_PRICE_SET_PRICE_LIST, see "Committing Price List Data to the BRM Database".

After your pricing tag and date range combinations expire, you can remove them from an **/offering_override_values** object by running the **pin_clean_offer_override** utility. The utility removes pricing tag and date range combinations by event type or date. For more information, see "pin_clean_offer_override".

29

Rating by Date and Time with Pipeline Manager

This chapter describes how to set up time models for the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Topics in this document:

About Rating by Date and Time with Pipeline Manager

A time model defines a combination of time periods. Each time period covers a specific time range for one or more days.

You use time models and time periods to charge different prices for the same service depending on the day and time.

Each time period contains the following information:

- **Day code**: The day or days the time period covers. For example, **Weekdays (Mon.-Fri.)**.
- **Time interval**: The time range the time period covers for the day code. For example, **08:00-17:00 Peak Time**.

> **Note:**
>
> A time model must cover all the days of the week. Within a time model, although you can create multiple day codes, you cannot use the same day in multiple day codes.

For example, one time model contains these time periods and is associated with pricing that charges for GSM telephony service as shown in Table 29-1:

Table 29-1 Rating by Date and Time

Time Period	Day Code and Time Interval	Charge from Pricing (Euro Cents per Minute)
WEEKPEAK - Weekdays Peak	Weekdays (Mon.-Fri.) 08:00 - 17:00	.15
WEEKOFF1 - Weekdays Off-Peak 1	Weekdays (Mon.-Fri.) 17:00 - 22:00	.12
WEEKOFF2 - Weekdays Off-Peak 2	Weekdays (Mon.-Fri.) 22:00 - 08:00	.10

Table 29-1 (Cont.) Rating by Date and Time

Time Period	Day Code and Time Interval	Charge from Pricing (Euro Cents per Minute)
WENDOFF - Weekend Off-Peak	Weekends (Sat., Sun., Holidays) 00:00-24:00	.09

> **Note:**
>
> Special days are not treated as normal days, so you can add special days to a time period without conflicting with any other days.

You include time models and time periods in charge configurations, which specify a pricing for a charge version. When a call is rated, the time model and time period, with the service code and impact category, determine the charge.

About Special Day Calendars

A special day calendar is a set of dates that you designate as being eligible for special rating. Each date can either be a specific date valid only in one year, such as May 11, 2003 for Mother's Day in the US, or a recurring date valid each year, such as January 1 for New Year's Day.

You include a calendar in a charge. The rating module uses the calendar to determine if a date in an event detail record (EDR) is a special day. You can include the same calendar in multiple charges, but you can also create different calendars for different charges.

Configuring Date and Time Rating

To configure date and time rating, do the following:

1. Create time models in Pricing Center or PDC and link them to charge configurations.
2. Create special day calendars in Pricing Center or PDC and link them to charges.
3. Configure the DAT_TimeModel and DAT_Calendar modules.

> **Note:**
>
> When you update time model and calendar data, you must use the **Reload** semaphore to reload data into the DAT_TimeModel and DAT_Calendar modules.

About Special Day Rates

Use special day rates to create special rates for specific times or days, independent of time models. You can create two types of special day rates:

- **Global**: Special day rates that apply to all accounts in your system. See "Setting Up Global Special Day Rates".

- **Account-specific**: Special day rates that apply only to one or more specific accounts in your system. You must create an extended rating attribute (ERA) for each account for this type of special day rate.

  See "Setting Up Account-Specific Special Day Rates".

> **Note:**
>
> When you create a special day discount, such as a birthday discount, any usage that begins on the discounted day and continues into the next, non-discounted day is fully discounted. Discounting does not split the charge packet at midnight but applies the discount for the entire call. For example, a customer receives a discount for 10% off on all birthday calls. On his birthday, the customer makes a 40 minute call that starts at 11:50 p.m. and ends on 12:30 a.m. the next day. The entire 40 minutes is discounted 10%.

Setting Up Global Special Day Rates

To set up special day rates that apply to all accounts, do the following:

1. Define special day rates in Pricing Center or PDC.
2. Configure the FCT_Dayrate module to calculate charges for date-based rates.
3. Configure the DAT_Dayrate module that stores the data used by the FCT_Dayrate module.

> **Note:**
>
> When you update special day rate data, you must use the **Reload** semaphore to reload data into the DAT_Dayratemodule.

Setting Up Account-Specific Special Day Rates

To set up special day rates for a single account or group of accounts:

1. Create an ERA for the special day.

   On the **Promotion** tab, select Special Day and enter the name BIRTHDAY and a date. For example, when creating a special day rate for a birthday, enter the date of the subscriber's birthday.

2. In Pricing Center or PCC, create a usage type that corresponds to the special day.

   For example, for a birthday special rate, you might create a usage type with a code of BIRTH.

3. In Pricing Center or PDC, include the usage you defined in a discount.

   You specify the usage type as part of the discount filter. The discount also specifies the amount of the discount for this rate.

4. Configure the IRL_UsageType iRule to map this usage type to EDR data.

   For example, for the usage type BIRTH, add a line similar to the following to the **IRL_UsageType.data** file:

   ```
   N;N;.;.;.;.;.;.;.;.;.;.;Y;.;.;BIRTH
   ```

30

Real-Time Rating Based on Event or Resource Quantity

This chapter describes how to set up quantity-based rating in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Before you read this chapter you should be familiar with rating and price lists. See "About Creating a Price List".

About Quantity-Based Rating

You use quantity-based rating when you must rate events based on previous rating or on the current balance of a resource. For example:

- $1 per fax for the first 10 faxes.
- 50 cents per fax for the next 90 faxes, up to and including the 100th fax.
- Free faxes for 101 or more.

To set up quantity-based rating, you group together balance impacts into *quantity discount brackets*, which are applied for a range of event quantities. For each range, you apply a different bracket group of balance impacts and you charge customers differently for each event quantity.

> **Note:**
>
> Quantity-based rating applies only to events rated in real time. You can use discounts to achieve many of the same results for both events rated in real time and events rated in batch by the pipeline.

Selecting a Quantity Discount Bracket Based on Previous Rating

You can use the event quantity as the basis for selecting a quantity discount bracket by using the Continuous option or the Rate Dependent option. When BRM requires only one rate for an event, the Continuous or Rate Dependent options do the same thing. When BRM requires

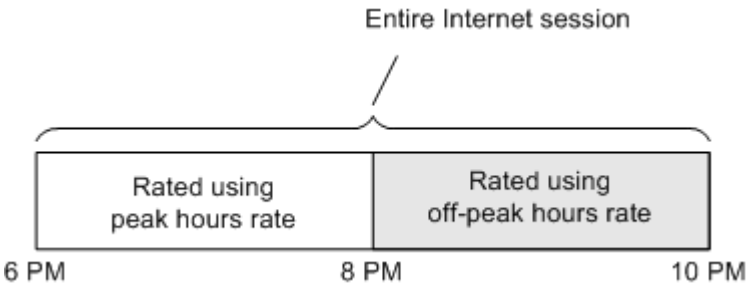
more than one rate for a given event, the difference between Continuous and Rate Dependent becomes important. For example, see the rates for Internet access shown in Table 30-1:

Table 30-1 Connection Times and Discount Calculations

Peak Hours Rate: 6 a.m. - 8 p.m.	Off-Peak Hours Rate: 8 p.m. - 6 a.m.
Quantity Discount Bracket 1: No min-No max Balance Impact: $1 per hour	Quantity Discount Bracket 1: 0-2 hours (first two hours) Balance Impact: $0.50 per hour
Quantity Discount Bracket 1: No min-No max Balance Impact: $1 per hour	Quantity Discount Bracket 2: 2+ hours (subsequent hours) Balance Impact: $0.25 per hour

In the example shown in Table 30-1, your customer connects to the Internet at 6 p.m. and disconnects at 10 p.m. The 2 hours between 6 p.m. and 8 p.m. are rated by the peak hours rate. The 2 hours between 8 p.m. and 10 p.m are rated by the off-peak hours rate. Figure 30-1 shows a timeline for this example.

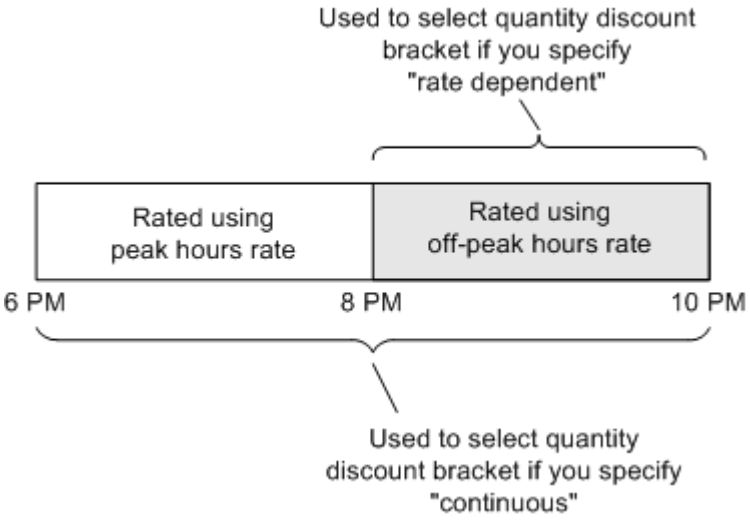
Figure 30-1 Internet Session Spanning Peak and Off-Peak Rates



Which quantity discount bracket from the off-peak hours rate is used depends on whether you specify *Continuous* or *Rate Dependent* as shown in Figure 30-2.

- Specify *Continuous* if the quantity used to select the quantity discount bracket for the off-peak hours rate will include *both* the quantity rate by using the peak hours rate and the quantity rated by using the off-peak hours rate.
- Specify *Rate Dependent* if the quantity used to select the quantity discount bracket for the off-peak hours rate will include *only* the quantity rated by using the off-peak hours rate.

Figure 30-2 Continuous Versus Rate Dependent Discounting



> **Note:**
>
> The **Resource Balance** option isn't applicable because you aren't basing the charges on a resource balance in the customer's account.

Selecting a Quantity Discount Bracket Based on a Resource Balance

When you specify *Resource Balance*, BRM selects a quantity discount bracket based on the balance of a resource. For example, by using the following rates in Table 30-2 to charge your customers for IP telephony:

Table 30-2 Rating Customers for IP telephony

Peak Hours Rate: 6 a.m. - 8 p.m.	Off-Peak Hours Rate: 8 p.m. - 6 a.m.
Quantity Discount Bracket 1: 0-10 hours of Internet time Balance Impact: $0.10 per minute	Quantity Discount Bracket 1: 0-10 hours of Internet time Balance Impact: $0.05 per
Quantity Discount Bracket 2: 10+ hours of Internet time Balance Impact: $0.07 per minute	Quantity Discount Bracket 2: 10+ hours of Internet time Balance Impact: $0.02 per hour

Between the hours of 6 a.m. and 8 p.m., the peak hours rate is used. If your customer has accumulated fewer than 10 hours of Internet time, your customer is charged $0.10 per minute for IP phone calls. If your customer has accumulated more than 10 hours of Internet time, your customer is charged $0.07 per minute for IP phone calls.

Between the hours of 8 p.m. and 6 a.m., the off-peak hours rate is used. If your customer has accumulated fewer than 10 hours of Internet time, your customer is charged $0.05 per minute for IP phone calls. If your customer has accumulated more than 10 hours of Internet time, your customer is charged $0.02 per minute for IP phone calls.

> **Note:**
>
> Resource balance-based rating has limitations for cycle events. If a cycle event is canceled at the end of a quantity discount bracket, BRM bases the cancellation fees on the next quantity discount bracket. For example, if the pricing configuration includes different cycle fees based on the number of elapsed months, you can create a counter for the number of elapsed months: quantity discount bracket 1 for months 0 to 6 (the first six months), and quantity discount bracket 2 for months 6+. If a customer cancels during the sixth month, the cancellation fee is based on quantity discount bracket 2.

Example 1: Rating Based on the Quantity of an Event

To rate events based on their quantity or duration, use either the Continuous (total quantity) or Rate Dependent option. For example, an IP session event uses the following 3 quantity discount brackets to rate the duration of the session:

1. First 10 minutes = $0.12 per minute
2. 10 - 20 minutes = $0.07 per minute
3. Over 20 minutes = $0.05 per minute

Notice that the maximum value in the first two brackets is the same as the minimum value in each subsequent bracket. Always set up balance impact groups this way to avoid errors in your price list.

With continuous rating, the maximum value does not represent an actual quantity; but rather a boundary between ranges that is not actually rated. In this example, the smallest fraction of time less than 10 minutes is rated in the first tier, and the smallest fraction of time greater than 10 minutes is rated by the second tier. Everything up to the maximum value in a range is included in that range, and anything over the maximum value is included in the next range.

Example 2: Rating Based on Total Event Quantities

This example shows how to rate the total number of hours per month for an IP session by using two quantity discount brackets.

- Quantity Discount Bracket 1: 0-10 hours a month = $0.10 a minute
- Quantity Discount Bracket 2: 10 + hours a month = $0.05 a minute

Set up the quantity discount brackets based on the resource balance “ISDN bulk hours". You base charges on the minutes per session and the monthly hourly total. Each bracket requires 2 balance impacts: 1 to track the charge per minute and 1 to track each hour of usage.

> **Note:**
>
> Because you track the charges based on the monthly total, you also require an ISDN bulk hours fold event that removes the total hours balance each month.

Example 3: Rating Based on Total Number of Events

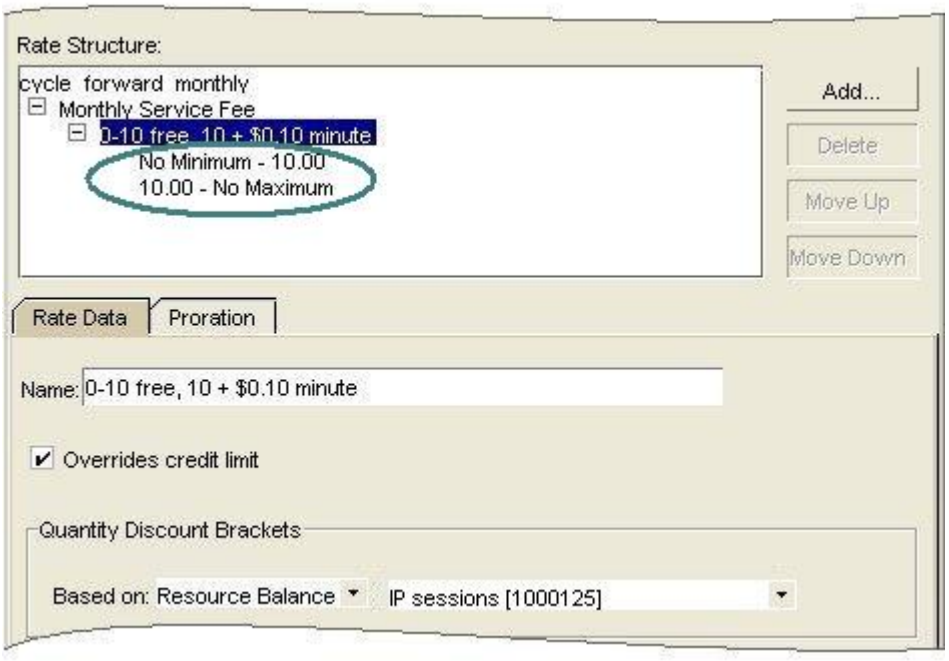
To rate events based on the total number of events that occur over a period of time, use the resource balance for that event. For example, if you charge customers for the total number of IP sessions they have each month, create a custom resource “IP_Sessions" and use it to track the total number of events each month.

For example, an IP Session event is rated based on the number of sessions by using the following 2 quantity discount brackets:

- Quantity Discount Bracket 1: 0-10 sessions = no charge
- Quantity Discount Bracket 2: more than 10 sessions = $0.10 per minute.

Figure 30-3 shows how to set up the brackets in Pricing Center:

Figure 30-3 Rate Structure Based on Number of Events



In the above example, sessions 1-10 are free. Each subsequent session is rated at $0.10 per minute. When the 11th session begins, the resource “IP Sessions" is incremented and the second balance impact containing the scaled amount ($0.10 per minute) is applied to the account balance. This way, when the next session occurs, the second rate is used.

> **Note:**
>
> Scaled amounts are applied to the account as soon as the event occurs. Fixed amounts are applied to the account after the event occurs.

The following examples show how to set up the balance impacts for each bracket.

Figure 30-4 shows the first discount bracket.

Set up the first bracket “No Minimum - 10.00" with one fixed amount impact to track the sessions. A scaled dollar impact is not necessary because there is no charge for the first 10 sessions.

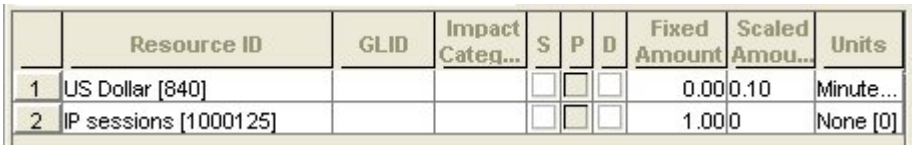
Figure 30-4 Quantity Discount Bracket 1

	Resource ID	GLID	Impact Categ...	S	P	D	Fixed Amount	Scaled Amou...	Units
1	IP sessions [1000125]	undefin...					1.00	0.00	None [0]

Figure 30-5 shows the second discount bracket.

The second bracket “10.00 - No Maximum" requires two balance impacts; one to charge the customer $0.10 per minute and one to track IP sessions.

Figure 30-5 Quantity Discount Bracket 2



	Resource ID	GLID	Impact Categ...	S	P	D	Fixed Amount	Scaled Amou...	Units
1	US Dollar [840]						0.00	0.10	Minute...
2	IP sessions [1000125]						1.00	0	None [0]

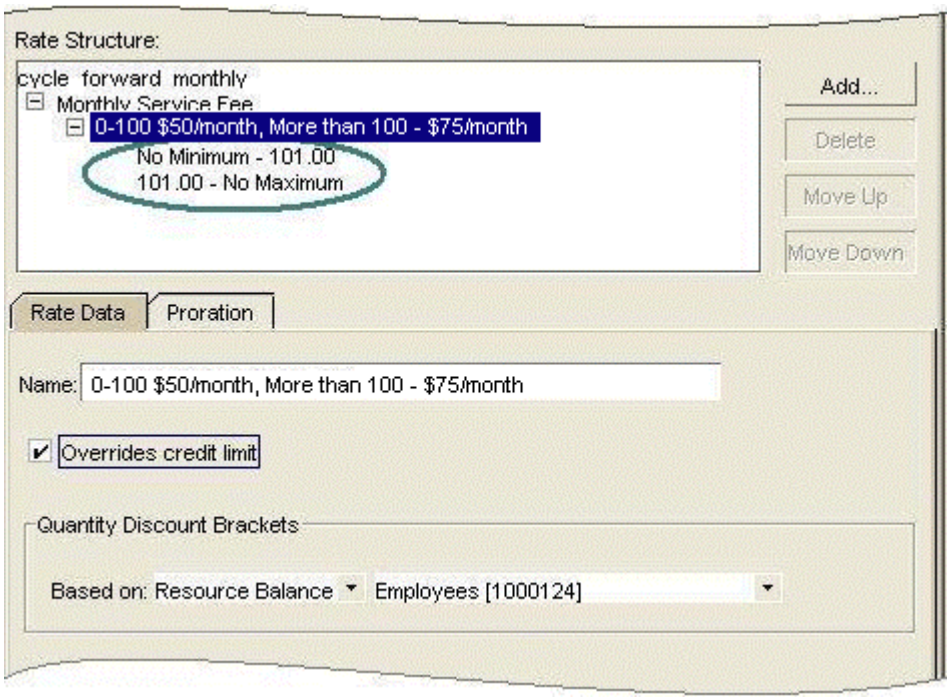
Example 4: Rating Based on a Resource Balance Not Affected by the Rated Event

You can charge customers based on resources that do not impact a customer's balance. For example, you can price IP usage based on the total number of employees in a company, or the net worth of a company. These resources are not directly related to the event you are rating. For example, if a company has 100 employees or less, you could charge them $0.03 per IP session; if they have more than 100 employees, you could charge them $0.01 per IP session.

> **Note:**
>
> Because the quantity discount brackets are based on a constant resource value, no upper boundary is included in the range. This is different from all other types of rating. In this example, the number of employees is independent of the number of IP sessions.

Figure 30-6 shows how to set up the brackets in Pricing Center:

Figure 30-6 Rate Structure Based on Number of Employees



Notice that the maximum value for the first bracket is 101 and the minimum value for the second bracket is 101. The maximum value is not included in the valid range and the minimum value is included.

Example 5: Rating an Event As If Using a Single Balance Impact Group

When you use a rate with multiple quantity discount brackets, which bracket is in effect changes as the event quantity changes.

For example, suppose you have the brackets to rate 15 hours of IP usage as shown in Table 30-3:

Table 30-3 Discount Brackets and Rating

Quantity Discount	Event Quantity	Fixed Amount	Scaled Amount	Result of Rating
Bracket 1: 0-5 hours	5	0	$1	$5.00
Bracket 2: 5-12 hours	7	0	$0.75	$5.25
Bracket 3: > 12 hours	3	0	$0.55	$1.65
NA	NA	NA	Total	$11.90

> **Note:**
>
> Because these fees are assessed on a per-hour basis, each bracket uses only a scaled amount.

Suppose, however, that you want to charge your customer as if all 15 hours were rated by the third group because *the total number of hours falls within that group's range*. In this case, the result is 12 hours @$0.55/hour or a fee of $8.25. To obtain this result, use fixed amounts in the second and third groups to compensate for what was charged in the previous groups as shown in Table 30-4:

Table 30-4 Rating Example

Quantity Discount	Event Quantity	Fixed Amount	Scaled Amount	Result of Rating
Bracket 1: 0-5 hours	5	0	$1	$5.00
Bracket 2: 5-12 hours	7	-$1.25	$0.75	$4.00
Bracket 3: > 12 hours	3	-$2.40	$0.55	-$0.75
NA	NA	NA	Total	$8.25

To determine the correct value for the fixed amount in each group, use the following formula:

$$F_c=-Q_p(S_p-S_c)$$

Where:

- Fc is the Fixed amount for current bracket.
- Qp is the maximum value of the Quantity range from the preceding bracket.
- Sp is the Scaled amount from the preceding bracket.
- Sc is the Scaled amount from the current bracket.

For example, determine the fixed amounts for the scenario above as follows:

Quantity Discount Bracket 1

A fixed amount is unnecessary.

Quantity Discount Bracket 2

Scaled amount for 5-12 hours = .75

Use the formula above and values from the previous bracket to determine the correct fixed amount:

$$F_c=-5(1-.75)]$$
$$F_c=-1.25$$

Quantity Discount Bracket 3

Scaled amount for more than 12 hours = .55

Use the formula above and values from the previous bracket to determine the correct fixed amount:

F_C=-12(.75-.55)]
F_C=-2.4

31
Setting Up Zones for Batch Pipeline Rating

This chapter describes how to set up zones for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Topics in this document:

About Zoning

Zoning adds call data to EDRs that can be used for rating. A *zone* is a combination of origin and destination area codes. You create zones to rate calls based on their origin and destination. You can also define zones for different services. For example, you can create zones for rating:

- Local calls.
- National calls.
- International calls.
- Wholesale rating of roaming usage.
- National calls for specific services such as SMS.

There are two types of zones:

- *Standard zones*. These are determined by evaluating the area code of the calling number (the A number) and the called number (the B number). See "About Standard Zones".
- *Geographical zone*s. These are determined by evaluating the distance between the origin and destination of the calls. See "About Geographical Zones".

All zones are mapped to impact categories. You combine impact categories with service usage data to create the final impact categories used by Pipeline Manager as the primary rating attribute. For more information, see "About Impact Categories".

When you configure zones in PDC, you configure the following:

- Standard zones.
- Geographical zones.
- APN maps (for rating GPRS events).
- Zone models, which are collections of zones.
- Impact categories.

About Impact Categories

You use Pipeline Manager impact categories to rate calls based on event attributes such as call origin and destination, and service attributes such as call forwarding. For example, you can create impact categories to rate:

- Local calls.
- Long distance calls.
- Local call forwarding.
- Long distance call forwarding.
- Local mailbox inquiries.
- National mailbox inquiries.

You create an impact category for each zone and usage scenario that you define. A usage scenario describes the type of call, for example, a long-distance mailbox inquiry. You map the following data to impact categories:

- In zone mapping, a zone and service code. The impact category and zone name are the same.
- In usage scenario mapping, a service code, service class, usage class, usage type, and zone.

You must define all possible impact categories before you can use them in a zone model. Before defining impact categories, it is helpful to know the following:

- How many and what kinds of zones you must configure, for example:
  - Domestic zones and international zones.
  - Zones for rating all wireless usage types or for specific usage types only, such as telephony, fax, and SMS.
  - Zones for rating retail usage and roaming usage.
  - No-charge zones such as toll-free and emergency destinations.
- How many and what kind of usage scenarios you must configure, for example:
  - Mailbox inquiries
  - Conference calling
  - Call forwarding
  - Family and friends calls

Your impact category names should reflect the usage scenario. For example, if you offer service-level agreements, you might create three impact categories called **Bronze**, **Silver**, and **Gold**.

For more information on creating usage scenarios, see "Setting Up Usage Scenario Mapping".

About Wholesale and Retail Impact Categories

An impact category can be used for either wholesale rating or retail rating. *Retail rating* applies to usage by your subscribers on your network. *Wholesale rating* applies to usage by visiting subscribers, for example, roaming calls.

You define impact categories according to your retail and wholesale needs. However, when you create impact categories, you do not specify whether they are used for wholesale or retail rating. Instead, you enter the impact category name in the wholesale and retail impact categories or zone fields when configuring zones and usage scenarios. You must enter values for both wholesale and retail impact categories and zones. When calls are rated, the type of impact category used is determined by whether the charge type is wholesale or retail.

About Standard Zones

A *standard zone* is a combination of an origin and destination area code and one or more services such as telephony and SMS. You assign an impact category to each zone that you define. For example, in Table 31-1, telephony origin and destination area codes are mapped to one of three impact categories; **Local**, **Toll**, and **LongDistance**:

Table 31-1 Impact Categories and Destination Area Codes

Origin Area Codes	Destination Area Codes	Destination Area Codes	Destination Area Codes
--	0123	0234	0345
0123	**Local**	**Toll**	**LongDistance**
0234	**Toll**	**Local**	**LongDistance**
0345	**Toll**	**LongDistance**	**Local**
0456	**LongDistance**	**LongDistance**	**Toll**

About Geographical Zones

A *geographical zone* is the distance between the origin and destination of a call. You set up geographical zones to rate calls by distance. This is useful when:

- Customers are located very near the border between two area codes. For example, if a customer in one area code calls a person two blocks away in another area code, you do not want to charge for a long-distance call.
- The distance covered by an area code is very large and you want to use several rates within the same area code. You do this by mapping sets of latitude and longitude coordinates to the same area code in a geographical model.

To set up geographical zones:

- Define geographical zones and map them to distances.
- Create geographical models to:
  - Define the area codes that make up each zone.
  - Map latitude and longitude coordinates to area codes.

Geographical mapping uses the area codes and geographic coordinates to compute the distance and assigns the zone that matches that distance to the call record.

Displaying Zoning Information on an Invoice

You can enter a zone description that can be included in invoices. To do so, in Pricing Center, enter the zone description in the **Description** field in the following dialog boxes:

- Standard Zone

- Geographical Zone
- Usage Scenario Mapping

You can use a maximum of 2,000 characters in the description.

> **Note:**
>
> To include zone descriptions on your invoices, you must configure the DAT_Zone and the FCT_USC_Map modules to load invoice description data into memory.

About Pipeline Manager Zone Modules

Zoning is performed by the following function and data modules:

- The FCT_APN_Map module adds Access Point Name (APN) data for the following:
  - To define zones.
  - To adjust impact categories.

  See "Setting Up APN Mapping".
- The FCT_PreRating module calculates zones and creates impact categories. This is the primary zoning module. See "Setting Up Prerating".
- The FCT_USC_Map module adjusts zones based on service attributes. See "Setting Up Usage Scenario Mapping".
- The DAT_Zone module handles zoning data for the zoning function modules.

The following function and iScript modules are used for optional zoning features:

- The FCT_Zone module calculates zones when you run Pipeline Manager for real-time zoning.
- The FCT_SegZoneNoCust maps zone models to segments. Use segment rating for collecting business information, for example, comparing the charges for the same events by using different charges.
- The FCT_MainZoning performs zoning for segment rating. It is used when a pipeline is used only for zoning.
- The ISC_SetSvcCodeRTZoning iScript updates DETAIL.INTERN_SERVICE_CODE EDR field with the customized service code value when different service codes are mapped to the same service type in the pipeline database and a specific service code is used in the zoning table.

Setting Up APN Mapping

Use the FCT_APN_Map module to rate GPRS events. The module runs before zoning or after zoning:

- **Before zoning**. If you rate calls that access a GPRS data service, for example, to download content, there is no B number. In that case, the module runs before zoning. It maps the access point name (APN) to a packet data protocol (PDP) which serves as the B number that can be used for zoning.
- **After zoning**. If you rate GPRS calls made from one phone to another, for example, to exchange calendar appointments, the EDR includes the A number and B number. In this

case, the zoning module has already created an impact category from the A number and B number. The FCT_APN_Map module uses the access point name (APN) to modify the wholesale and retail impact categories.

To configure APN mapping, you do the following:

1. Create APN maps. See "About APN Maps".
2. Configure the FCT_APN_Map module.

About APN Maps

To map APN numbers to PDPs before zoning or to impact categories after zoning, the FCT_APN_Map module reads data from the EDR and evaluates it according to the APN map. An APN map includes one or more mapping rules that specify the data that must be matched to apply the mapping.

You can create multiple sets of APN mapping rules, for example, different mappings for wholesale and retail rating.

You can use APN groups to assign a set of rules to a specific FCT_APN_Map module instance.

You can use the following EDR data to create an APN mapping rule:

- These values are compared with the EDR data when the module runs before or after zoning:
  - APN name (from the Associated GPRS Extension record APN_ADDRESS field)
  - Internal service code
- These values are compared with the EDR data only when the module runs after zoning:
  - Wholesale zone
  - Retail zone

You rank APN mapping rules when you create them. The first rule that matches the EDR data is used. The results of the mapping add the following data to the EDR:

- If the module runs before zoning, it adds the PDP address in the detail record INTERN_B_NUMBER_ZONE field.
- If the module runs after zoning, it adds the following:
  - Impact category
  - Wholesale zone
  - Resale zone

When you run the FCT_APN_Map module after zoning, different EDR fields are used depending on whether you use FCT_PreRate for zoning and rating, or FCT_Zone for zoning only:

- When FCT_PreRate is used (for zoning and rating):
  - The impact category is read from and written to the associated charge breakdown record.
  - The wholesale and retail zones are read from and written to the supplementary zone packet record.
- When FCT_Zone is used (for zoning only), the wholesale and retail zones are read from and written to the detail block of the EDR.

The following tables summarize the data mappings.

Table 31-2 describes the EDR field mapping before zoning when there is no B number.

Table 31-2 Running FCT_APN_Map before Zoning

EDR Field	Compared against This Pipeline Manager Database Field
AssocCharge-APN group	IFW_APN_MAP.APN_GROUP
Detail-Internal Service Code	IFW_APN_MAP.SERVICECODE
Detail-Application	IFW_APN_MAP.ACCESSPOINTNAME

Running FCT_APN_Map after zoning:

For every zone breakdown record the following mapping is performed as shown in Table 31-3:

Table 31-3 Running FCT_APN_Map after Zoning

EDR Field	Compared against This Database Field
AssocZone-APN group	IFW_APN_MAP.APN_GROUP
Detail-Internal Service Code	IFW_APN_MAP.SERVICECODE
Detail-Application	IFW_APN_MAP.ACCESSPOINTNAME
AssocZone-Wholesale Zone Result value	IFW_APN_MAP.ZONE_WS
AssocZone-Retail Zone Result value	IFW_APN_MAP.ZONE_RT

Creating APN Maps

Use PCC to create APN maps and APN groups. APN groups are assigned differently depending on whether you run the FCT_APN_Map module before or after zoning:

- **Before zoning**. Specify the APN map in the **APNGroup** registry setting.
- **After zoning**. Specify the APN map in the zone model.

When you create mapping rules, you use regular expressions.

Setting Up Prerating

Prerating is performed by the FCT_PreRating module. This module uses the data in the EDR to compute zones and assign an impact category to each charge packet.

The module works as follows:

1. It finds the charge to use by reading the charge code in the charge packet.

> **Note:**
>
> When an EDR is associated with multiple charges, FCT_PreRating uses the highest priority charge. That is, it searches through all associated charges, from highest priority to lowest priority, until it finds a charge offer that matches the event's criteria. When two charges have matching priority, FCT_PreRating chooses the charge with the earliest start time.

2. In the charge, it finds the zone model.
3. It uses the zone model to calculate the zone and write the impact category to the charge packet. It reads the following information in the EDR:
   - Service code
   - A number
   - B number
   - Call start timestamp

   The data used for determining zones and impact categories is handled by the DAT_Zone module.

To set up prerating, you do the following:

1. Create zone models and impact categories in PDC.

   You can also define zone data for the DAT_Zone module in ASCII files. See "Creating Zone Data Files".

   > **Note:**
   >
   > You can use DAT_Zone in file mode in an FCT_Zone/DAT_Zone combination only. Because file-based zoning cannot coexist with pre-rating, DAT_Zone in file mode does not work with FCT_PreRating.

2. Configure the FCT_PreRating module.
3. Configure the DAT_Zone module.

Creating Zone Data Files

Zone data can be defined in ASCII files.

- Each line in a file contains one configuration entry.
- Fields are separated by semicolons (;).

Master Data File

Master data files defines the zone model codes in the format:

```
ZoneModelCode;ZoneModelType;GeoModel
```

where:

ZoneModelCode specifies a unique code for the zone model.

ZoneModelType specifies the zone model type; standard (**S**) or geographical (**G**).

GeoModel specifies the geographical link.

Master data file example:

```
ZM_ADD;S;100000
ZM_GEO;G;100000
```

Standard Zone File

Standard zone file defines standard zone models in the format:

```
ServiceCode;ValidFrom;ValidTo;Origin;OriginDesc;Destination;DestinationDesc;RtZone;WsZone
;ZoneEntryName;ZoneDescription;AltZoneModel
```

where:

ServiceCode is the service code that applies to the zone.

ValidFrom is the date when the standard zone becomes valid, in the format DD.MM.YYYY. For example, **31.12.2004**.

ValidTo is the date when the standard zone becomes inactive. Use the same date format as above.

Origin is the origin area code for the zone.

OriginDesc is the description for the origin.

Destination is the destination area code for the zone.

DestinationDesc is the description for the destination.

RtZone is the impact category for retail rating to apply to the zone.

WsZone is the impact category for wholesale rating to apply to the zone.

ZoneEntryName is the name for the standard zone.

ZoneDescription is the description for the standard zone.

AltZoneModel is the alternate standard zone model that can be used.

> **Note:**
>
> The *ZoneEntryName* and *ZoneDescription* fields must be included, even if they are not used.

Standard zone file example:

```
TEL;20030101;;0049;Germany;0049;Germany;NAT_MBI;NAT_MBI;;;
```

Geographical Zone File

Geographical zone file defines geographical zone models in the format:

```
Distance;ServiceCode;ValidFrom;ValidTo;WsZone;RtZone;ZoneEntryName;ZoneDescription;AltZon
eModel
```

where:

Distance is the maximum distance for the geographical zone.

ServiceCode is the service code that applies to the zone.

ValidFrom is the date when the zone becomes valid, in the format DD.MM.YYYY. For example, **31.12.2004**.

ValidTo is the date when the zone becomes inactive. Use the same date format as above.

WsZone is the impact category for wholesale rating to apply to the zone.

RtZone is the impact category for retail rating to apply to the zone.

ZoneEntryName is the name for the geographical zone.

ZoneDescription is the description for the geographical zone.

AltZoneModel is the alternate geographical zone model that can be used.

> **Note:**
>
> The *ZoneEntryName* and *ZoneDescription* fields must be included, even if they are not used.

Geographical zone file example:

```
500;*;19990101;;NAT_FIX;NAT_FIX;;;
```

Area Code Coordinate File

Area code coordinate file defines area codes used in a geographical zone in the format:

```
GeoModel;AreaCode;Longitude;Latitude;ValidFrom;ValidTo
```

where:

GeoModel is the description for the geographical link.

AreaCode is the area code to link.

Longitude is the longitude for the area code.

Latitude is the latitude for the area code.

ValidFrom is the date when the geographical link becomes valid, in the format DD.MM.YYYY. For example, **31.12.2004**

ValidTo is the date when geographical link becomes inactive. Use the same date format as above.

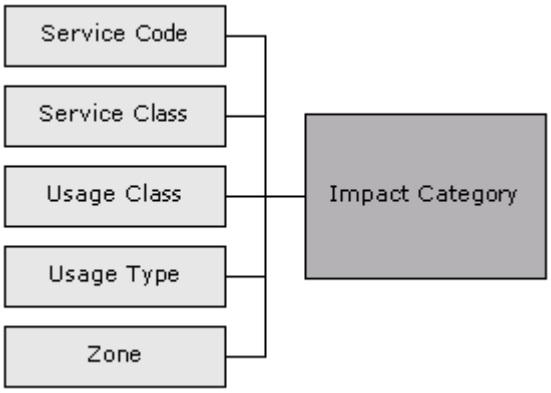
Area code coordinate file example:

```
100000;049;80;80;19990101;19990115
```

About Usage Scenario Mapping

Usage scenario mapping brings together the data in all EDR fields pertaining to the service code, usage class, usage type, and zone, and then adds the final impact category used for rating the EDR to the charge packet or the detail block, depending on which USC mapping mode you selected as shown in Figure 31-1.

Figure 31-1 Usage Scenario Mapping Factors



The impact category is included as part of the charges.

You define usage scenario mapping to rate differentiated network services, such as mailbox inquiries and friends-and-family discounts.

For example:

- If a customer has a mailbox service, you can create separate impact categories for accessing a mailbox from your network or from a different carrier's network. In this example, the impact category uses two values, the usage class that represents the mailbox, and the zone that the event applies to.
- If you provide a friends-and-family discount, you can create separate impact categories for national friends and international friends.
- If a customer has call forwarding, you can create separate impact categories for calls forwarded to a local number or to a long-distance number.

You can use multiple attributes to define an impact category, or you can use only one attribute.

> **Note:**
>
> You can use iScript to define your own attributes to be considered when specifying the final impact category.

Setting Up Usage Scenario Mapping

To set up usage scenario mapping, do the following:

1. Create usage scenario maps. You have two options:
   - You can create usage scenario maps in PCC. In this case, the usage scenario map data is stored in the Pipeline Manager database.
   - You can create a file that defines usage scenario maps. In this case, the FCT_USC_Map module reads the file. For information on creating the file, see "Creating a Usage Scenario Map File".
2. Configure the FCT_USC_Map module.

You specify the mode in which USC mapping is done.

- The **Rating** mode (the default) specifies that USC mapping is done using the zone model from the charge packets. Mapping in this mode provides the impact category for charge packets.
- The **Zoning** mode specifies that USC mapping is done using the zone model from the EDR detail block. Mapping in this mode provides impact categories for the detail block.

3. Configure the DAT_USC_Map module.

   When you use the database interface, you create USC map groups in PCC, which contain the USC maps. You associate USC map groups with zone models. FCT_USC_Map uses the zone model ID in the EDR to identify the associated USC map group. The zone model ID is added to the EDR by the FCT_PreRating module. See "Setting Up Prerating".

> **Note:**
>
> If there is no zone value in the EDR, FCT_USC_Map writes an empty string as the value of the zone to the output EDR. This should be considered when you set up the mapping.

> **Note:**
>
> You can use semaphore commands with the DAT_USC module to save USC Map data to a file.
>
> - Use **PrintAllUscMapData** to save all of the USC Map data to a file.
> - Use **PrintUscMapDataForZoneModel** to save the data for a specified Zone Model ID.
> - Use **PreCompiledDataDir** to compile USC mapping data and save the data to a specified directory.
> - Use **NumberOfThreads** to specify the number of threads to use when compiling and loading the mapping data.
> - Use **UscGroups** to specify the USC groups for which to load rules.

About Usage Scenario Maps

Usage scenario mapping is performed by the FCT_USC_Map module. To define the final impact category, the module reads data from the EDR and evaluates it according to the usage scenario map. A usage scenario map includes one or more mapping rules that specify the data that must be matched to apply the impact category.

When you create usage scenario maps, you create a mapping rule for each impact category. You can also define the order in which the rules are evaluated. The impact category is derived from the first rule that matches the data. If no matching rule is found in the usage scenario mappings, then no mapping is performed.

When no usage scenario mappings are set up, FCT_USC_Map module uses the default USC map group. To use a default USC map group, you must create it and specify the default USC map group in the FCT_USC_Map registry.

You can use the following data to create a mapping rule:

- The EDR date and time. This data is read from the EDR **StartTimeStamp** field.
- Maximum event quantity, for example, 60 seconds. If the quantity exceeds this value, the map does not apply.
- Minimum and maximum wholesale amount.
- Usage class.
- Usage type.
- Service code.
- Service class.
- Wholesale and retail impact category.

When you create the mapping rules, you use regular expressions.

Creating a Usage Scenario Map File

The USC mapping rules are defined in an ASCII file.

- Each rule consists of a list of fields. The following table describes the meaning of each field.
- Every new line defines an adjustment rule.
- The position of each rule in the file determines its rank. The first matching rule is used.
- Fields are separated by semicolons (**;**).
- Comment lines start with #.
- Empty lines are allowed.

> **Note:**
>
> The value of the field rank is ignored. The evaluation order of the rules is given by the order of the rules within the file.

Table 31-4 lists the fields in the file:

Table 31-4 Fields in Usage Scenario Mapping File

Position	Identifier	Description
1	**USCGroup**	Specifies the compare pattern for the source USC group.
2	**Rank**	This parameter is not evaluated. Instead, the rules are evaluated in the order they appear in the file.
3	**ValidFrom**	Specifies the start date for the USC mapping rule. This can either be a date with the format YYYYMMDD or a weekday with an optional timestamp, for example: • 20030524 • SAT • MON16:00 If the field is left empty, the earliest possible date 19010101 is used.

Table 31-4 (Cont.) Fields in Usage Scenario Mapping File

Position	Identifier	Description
4	**ValidTo**	Specifies the end date for the USC mapping rule. This can either be a date with the format YYYYMMDD or a weekday with an optional timestamp. If the field is left empty, the latest possible date 20370205 is used.
5	**TimeFrom**	Specifies the beginning of the time period (in format HH:MM) the mapping entry is valid on each day of the validity period specified by **ValidFrom** and **ValidTo**. If the field is left empty, 00:00 is used.
6	**TimeTo**	Specifies the end of the time period (format HH:MM) the mapping entry is valid on each day of the validity period specified by **ValidFrom** and **ValidTo**. If the field is left empty, 24:00 is used. Example: To set up a mapping entry that is valid on weekends between 13:00 and 14:00, you must set **ValidFrom**=SAT, **ValidTo**=SUN, **TimeFrom**=13:00 and **TimeTo**=14:00.
7	**Quantity**	Specifies the maximum quantity value for an EDR container. If this maximum is exceeded, the mapping rule will not be used. If this field is left empty or a 0 is specified, the rule is valid for every quantity value. For example, to avoid mapping for calls longer than 60 seconds, set the value to **60**.
8	**MinAocAccount**	Specifies the optional minimal wholesale amount value (the wholesale charge). The map is valid only if the EDR charge value is greater or equal to this setting.
9	**MaxAocAccount**	Specifies the optional maximum wholesale amount value (the wholesale charge). The map is valid only if the EDR charge value is less than or equal to this setting.
10	**SourceUsageClass**	Specifies the compare pattern for the source usage class.
11	**SourceUsageType**	Specifies the compare pattern for the source usage type.
12	**SourceServiceCode**	Specifies the compare pattern for the source service code.
13	**ServiceClass**	Specifies the compare pattern for the source service class.
14	**SourceImpactCategoryWS**	Specifies the compare pattern for the source wholesale impact category
15	**SourceImpactCategoryRT**	Specifies the compare pattern for the source retail impact category.
16	**NewUsageType**	Specifies the resulting usage type for this rule. If this field is left blank or set to **0**, the usage type is not modified.
17	**NewImpactCategoryWS**	Specifies the compare pattern for the wholesale zone which should be mapped.
18	**NewImpactCategoryRT**	Specifies the compare pattern for the retail zone which should be mapped.
19	**Description**	Specifies the rule name. It is not used for rule evaluation.

Converting Zone Values for Output

Use an IRule script to convert the alphanumeric zone values (impact category name) into external result values before the EDR container is written to the output file. Be default, the IMPACT_CATEGORY values from IFW_IMPACT_CAT are written to the output file as the zone value.

Setting Up Multi-Segment Zoning

Use multi-segment zoning to test different zoning models against the same EDR data. Use the FCT_SegZoneNoCust module and the FCT_MainZoning module to perform zoning for multi-segment zoning.

Use the FCT_SegZoneNoCust module to perform zoning when you cannot get customer account data about the EDRs. For example, you might want to use a test system that does not have access to account data. In that case, you configure the FCT_SegZoneNoCust module to find the segment based on the source network ID.

After the FCT_SegZoneNoCust module finds the segment, it adds the zone data for that segment to the EDR. It adds one associated zone breakdown record for each zone in the segment. The FCT_MainZoning module uses this data to perform the zoning and add the results to the zone packets.

To configure multi-segment zoning, configure the FCT_SegZoneNoCust and FCT_MainZoning modules.

Configuring Segments in the FCT_SegZoneNoCust Module

Use the FCT_SegZoneNoCust module **Segments** registry entry to specify the segment to use for each source network. Each rule defines the connection between the source network and the segment. For example:

```
26201 = SegmentD1
26202 = SegmentD2
```

> **Note:**
>
> You cannot change these mappings during run time.

32

Real-Time Rating Using Event Attributes

This chapter describes how to define rates in your Oracle Communications Billing and Revenue Management (BRM) price list by using event attributes such as telephone call origins and destinations.

Topics in this document:

About Using Event Attributes to Rate Events

If BRM can measure a billable event, it can rate it. It can rate an event based on multiple event attributes, such as the date and time that the event occurred. With attribute-based rating, you can specify additional event attributes to consider for rating, for example, phone call origin and destination.

> **Note:**
>
> You can use a rate plan selector to rate cycle events. To do so, you can use account or service attributes, but not event attributes, to select the rate plan.

Using Event Attributes to Select Rate Plans

To use attribute-based rating, you use the Pricing Center *rate plan selector*. The rate plan selector associates a value in an event attribute with a rate plan. For example, by using the event attribute that stores the telephone call destination, you can choose rate plans like this:

- If the call is made to France, use the Calls to Europe rate plan.
- If the call is made to Ghana, use the Calls to Africa rate plan.

Figure 32-1 shows a rate plan selector that chooses a different rate plan depending on call type. In this case, the telephony event includes an attribute named CALL_TYPE, which specifies the type of call, for example, a call made with a calling card. To rate a call made with

a calling card differently from a call without a calling card, you assign a different rate plan based on the call type.

Figure 32-1 Rate Plan Selector Configuration for Call Type

Row	EVENT.PIN_FLD_CALL.PIN_FLD_TYPE	+...	Rate Plan	Impact Category
1	5		Calling card	default
2	1		Home phone	default

The event attributes that you use for rating often depend on the source of the usage data. For example, when rating telephone calls, some PBX-to-gateway/gatekeeper systems do not supply the ANI value. In that case, you can use the CALL.ORIG value instead.

Using Event Attributes to Define Impact Categories

You use impact categories to apply different balance impacts for the same rate plan. For example, you can assign a set of telephone calls and destinations to a single rate plan:

Rate plan: IP Telephony

Impact category: USA to Europe

Balance impact: .10

Impact category: USA to Asia

Balance impact: .15

Notice that the same rate plan is used for all calls, but the impact category specifies different balance impacts.

Even though you are using a single rate plan here, you must use the rate plan selector to assign impact categories. Therefore, you must choose **Rate plan selector** when creating the product, not **Single rate plan**.

You use the rate plan selector to associate event attributes with impact categories.

The event attributes ANI (call origin) and DNIS (call destination) are entered as country codes:

- 1 = USA
- 33 = France
- 49 = Germany
- 81 = Japan
- 86 = China

A call from the USA to France might have this call origin and destination:

- Origin: 15555121212
- Destination: 3372621234

BRM matches the numbers in the rate plan selector with the telephone numbers. Since only country codes are used, all calls from each country match the numbers in the rate plan selector. For example, “3372621234" matches “33."

In the balance impacts for this rate plan, the impact categories define how much to charge.

> **Note:**
>
> In **Balance Impacts**, you can use asterisk (*) as a wildcard character in the **Impact Category** column to apply the balance impact to all impact categories defined for the rate plan.

Using the above examples, here is how BRM assigns a balance impact to a call:

1. The customer makes a call from the USA to France.
2. During rating, BRM finds the call origin and destination data in the event.
3. BRM looks through the data in the rate plan selector to find entries that have a call origin of USA and a call destination of France. This information tells BRM which rate plan to use, and which impact category to use.
4. BRM looks in the IP Telephony rate plan for the balance impact.
5. BRM looks through the IP Telephony balance impacts to find one that uses the USA_Europe impact category. BRM uses that balance impact to calculate the charge for the call.

About the Default Impact Category

You use the default impact category to apply a balance impact when none of the other impact categories can be used. The default impact category provides a fail-safe; without it, some events might not be rated at all.

The default impact category name is **default**.

Setting the impact category to **default** applies the balance impacts of the rate plan according to standard rating attributes, and ignores all impact category conditions.

Creating Impact Categories

To make impact categories available to Pricing Center, you edit the **pin_impact_category** file, then run the **load_pin_impact_category** utility to load the contents of the file into the **/config/impact_category** object in the BRM database.

> **Note:**
>
> The **load_pin_impact_category** utility requires a configuration (**pin.conf**) files.

1. Edit the **pin_impact_category** file in *BRM_Home***/sys/data/config**. The **pin_impact_category** file includes examples and instructions.

> **Note:**
>
> The **load_pin_impact_category** utility overwrites existing impact categories. If you are updating impact categories, you cannot load new impact categories only. You must load a complete set of impact categories each time you run the **load_pin_impact_category** utility.

2. Save the **pin_impact_category** file.
3. Use the following command to load the file into the database:

   ```
   load_pin_impact_category pin_impact_category
   ```

   If you are not in the same directory as the **pin_impact_category** file, include the complete path to the file, for example:

   ```
   load_pin_impact_category BRM_Home/sys/data/config/pin_impact_category
   ```

   For more information, see "load_pin_impact_category".
4. Stop and restart the Connection Manager (CM). If necessary, stop and restart Pricing Center.

To verify that the impact categories were loaded, you can display the **/config/impact_category** object by using the Object Browser, or use the **robj** command with the **testnap** utility.

Creating Unassigned Impact Categories

You can include more impact categories in the **pin_impact_category** file than you need. You do not have to assign a balance impact to all impact categories.

For example, you can plan ahead for origin/destination pairs that you will add later and create the impact categories that they will use.

About Charging for Custom Events and Attributes

BRM supports a default set of event attributes, such as geographical zones, quality of service, and so on. You can define charges for these attributes by using the Rate Plan Selector.

If you want to rate custom attributes of events or non-event attributes or if your rating guidelines are more complex than the defaults provided with BRM, you must modify the PCM_OP_ACT_POL_SPEC_RATES policy opcode.

For example, you can rate administrative events or non-event attributes such as an account attribute or a profile attribute. You can modify the policy to rate percentage values or values containing a greater than or less than operator. You can rate an event one way if an attribute is greater than 10, and another way if it is less than 10.

Charging for Custom Events and Attributes

To charge for custom events and attributes:

1. Modify the PCM_OP_ACT_POL_SPEC_RATES policy opcode or write a custom policy opcode to analyze the event data and assign the rate plan and impact category.

   See the policy source file of the PCM_OP_ACT_POL_SPEC_RATES policy opcode for a sample implementation.

For information on how generic rating works, see the description of PCM_OP_ACT_USAGE.

2. Edit the **pin_spec_rates** file in *BRM_Home*/**sys/data/config** to associate the opcode to the event type that it rates.

   The **pin_spec_rates** file includes examples and instructions.

   > **Note:**
   >
   > When you run the **load_pin_spec_rates** utility, it overwrites the existing setup for administrative events charges. If you are updating a set of administrative events charges, you cannot load new charges only. You load complete sets of charges each time you run **load_pin_spec_rates**.

3. Create a configuration file for the "load_pin_spec_rates" utility.
4. Run the **load_pin_spec_rates** utility to load the contents of the **pin_spec_rates** file into the database.

   ```
   load_pin_spec_rates pin_spec_rates_file
   ```

5. Edit the **pin_impact_category** file in the *BRM_Home*/**sys/data/config** directory to define impact categories for the event.

   The **pin_impact_category** file includes examples and instructions.

   > **Note:**
   >
   > When you run the **load_pin_impact_category**, it overwrites the existing impact categories. If you are updating a set of impact categories, you cannot load new impact categories only. You load complete sets of impact categories each time you run the **load_pin_impact_category** utility.

6. Run the "load_pin_impact_category" utility to load the impact categories into the database.

   ```
   load_pin_impact_category pin_impact_category_file
   ```

7. If you use pipeline batch rating, add the custom cycle event to the following locations:
   - The EAI payload configuration file. See "About the Payload Configuration File".
   - Your system's event notification list. The information you must add to the list is in the **pin_notify_ifw_sync** file.
8. In Pricing Center, create the rate plans that can be used to rate an event that uses custom event analysis.
9. In your pricing plan, assign a rate for the impact category.
10. Use the Pricing Center to create products that use the rates defined in your custom opcodes. Ensure that the products use custom event analysis.

Using Custom Fields with Real-Time Rating

If you add rating-related custom fields to BRM, you must make these fields available to the rating opcodes. For instructions, see "Making Custom Fields Available to Your Applications" in *BRM Developer's Guide*.

About Real-Time Zoning

You use zones to group event attribute values into manageable categories. For example, to apply the same rate plan or impact category to all area codes in Northern California, you can create a zone that includes all Northern California area codes. Another zone might include all Southern California area codes. You then might include both zones in a zone for all California area codes, as shown in the following example:

California zone

Northern California

408

415

650

Southern California

213

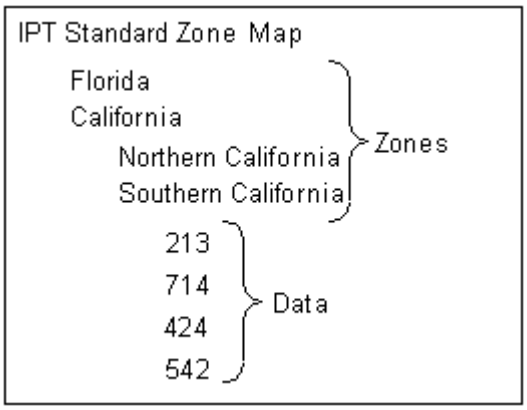
714

424

A zone map contains a hierarchy of zones and *data*, such as phone numbers, IP addresses, and Quality of Service (QoS) data.

Figure 32-2 shows a zone map with multiple nested zones. The data consists of area codes:

Figure 32-2 Zone Map with Nested Zones



If you use a zone map for an event attribute, all of that type of event attribute must be defined with a zone or data from that zone map. You cannot mix hard-coded values in a rate plan selector that uses zones.

For more information on zone, see "About Zoning".

About Setting Up Zones

When you use zone maps to rate events, BRM looks for the product to use for rating an event. If the product is configured to use a pipeline zone model, the real-time rating opcodes send the

event to a real-time zoning pipeline to retrieve the zoning information. If the product is not configured to use a pipeline zone model, the real-time rating opcodes search the BRM database for the zoning information for the event.

You can define zones to rate events by using Pricing Center or Zone Mapper:

- To use zone information for rating both real-time and batch rating, use Pricing Center. Zone maps defined using Pricing Center are stored in the Pipeline Manager database.

  If you use both real-time and batch processes for rating events, or if you are a new BRM user, define your zones by using Pricing Center.

  See "About Zoning" and "Setting Up Zones by Using Pricing Center".

- To use zone information for rating only real-time events, use Zone Mapper in Pricing Center. Zone maps defined by using Zone Mapper are stored in the BRM database.

  If you have already defined zone maps in BRM by using Zone Mapper, you can continue to use your existing zone maps.

  See "About Setting Up Zones by Using Zone Mapper ".

Pricing Center displays the zone-map names and impact categories from both the Pipeline Manager database and the BRM database. You can use the zone maps stored in either database to configure price plans.

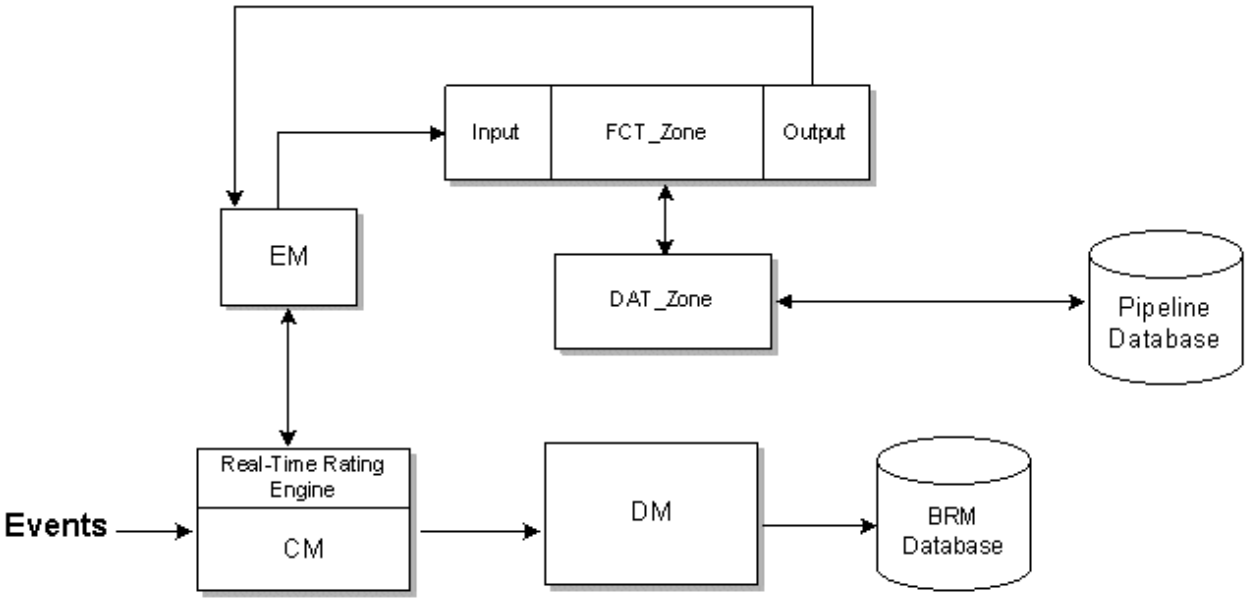
Real-Time Zoning Pipeline Architecture

A real-time zoning pipeline processes events as follows:

1. When the CM receives an event with a zoning request, the real-time rating opcodes check the event and the product that rates the event.
2. If the product is configured for pipeline zoning, the real-time rating opcodes extracts the calling number, called number, service calls, event time, and zone map name from the event and product flist.
3. BRM sends the event flist in a request to the NET_EM module to get the impact category from the Pipeline Manager database for the event. The CM sends the request for specific zone map data, such as standard zone-map data and geographic zone-map data. See "Configuring the NET_EM Module for Real-Time Processing".
4. When an event flist is passed to a real-time zoning pipeline, the input module in the pipeline converts the event data from the flist format used by real-time rating opcodes to the EDR format used in the pipeline.
5. The FCT_Zone module processes the zoning EDR and uses the DAT_Zone module to search for the zone map for the event.
6. When zoning is complete, the EDR, now containing zoning information for the event, is converted back into flist format by the output module in the pipeline. The output module also passes the event back to the real-time rating opcodes.
7. If a correct zone map for the event is not found in the Pipeline Manager database, the pipeline zoning module returns an empty flist to the CM. The CM logs the error and the event is not rated.

Figure 32-3 illustrates the data flow for real-time pipeline zoning:

Figure 32-3 Data Flow for Real-Time Pipeline Zoning



Setting Up Zones by Using Pricing Center

1. In Pricing Center, set up your zone models, which are stored in the Pipeline Manager database.

   > **Note:**
   >
   > Ensure that the zone map names are unique between both the BRM and Pipeline Manager databases.

2. Configure the NET_EM to receive events from the real-time rating opcodes.
3. Configure the CM to point to the NET_EM.
4. Configure the INP_Realtime module.

   Use this entry for the OpcodeMapping entry:

   ```
   OpcodeMapping = ./formatDesc/Formats/Realtime/zonemap_event.xml
   ```

5. Configure the OUT_Realtime module.
6. Configure the FCT_Zone module to specify a connection to the DAT_Zone module.
7. Configure the DAT_Zone module to process real-time events.

About Setting Up Zones by Using Zone Mapper

You use Zone Mapper to create zones and zone maps, which are stored in the BRM database. Zone Mapper is a component of Pricing Center. You can create multiple zone maps.

> **Note:**
>
> You can use a rate plan selector to rate cycle events. To do so, you can use account or service attributes, but not event attributes, to select the rate plan.

> **Note:**
>
> A zone map name can include a maximum of 255 characters.

About Defining Zone Maps for Products

Each product can have only one zone map. However, one zone map can be used by multiple products. You can create multiple zone maps for one system or for branded systems.

> **Note:**
>
> All brands in a database can use any of the zone maps stored in that database.

Options for Matching Event Attribute Values

When you create zones using Zone Mapper, you specify how BRM matches a value in an event with the data in the zone map. You have two choices:

- **Prefix:** BRM searches for the value in the zone map that best matches the event. For example, in an IP telephony service, if the customer calls 202-555-5678, and the zone map includes 202, BRM will find the zone with 202. If the zone map includes both 202 and 202-555, BRM will find the zone that includes 202-555.
- **Exact Match:** BRM searches only for the value in the zone map that exactly matches the event. For example, if the customer calls 202-555-5678, BRM looks only for a zone map that includes 202-555-5678.

Changing and Deleting Zone Maps

You can either change a zone name, modify the data for a zone, or delete a zone map entirely. When you delete a zone map that you used to define a product in Pricing Center, you must also remove the map from that product. When you delete a zone map, you also delete the zones that belong to it.

> **Note:**
>
> - Deleting a zone map permanently removes the map from the BRM database.
> - If customers have already purchased a product that includes a zone map that you are deleting, you must cancel the product that includes the old zone map, and purchase the product that includes the new zone map.

Selecting Multiple Rate Plans and Impact Categories

You can use the rate plan selector to choose rate plans and assign impact categories at the same time.

Figure 32-4 shows a rate plan selector that selects rate plans based on three event attributes:

- Call origin
- Call destination
- Call type

Figure 32-4 Rate Plan Selector

Row	D_CA...	D_CAL...	EVENT.PIN_FLD_CALL.PIN_FLD_TYPE	+...	Rate Plan	Impact Category
1	US	Europe;France	5		Calling card	US_to_Europe
2	US	Europe;Germany	5		Calling card	US_to_Europe
3	US	Europe;France	1		Telephony	US_to_Europe
4	US	Europe;Germany	*		Telephony	US_to_Asia

Customizing Zone Mapping

The zone mapping opcodes perform zoning for real-time rating.

How Zone Mapping Works

To manage zones, for example, create, delete, and update zone maps, use PCM_OP_ZONEMAP_COMMIT_ZONEMAP. Zone maps are stored in **/zonemap** objects.

By default, PCM_OP_ZONEMAP_COMMIT_ZONEMAP takes an array of zone maps as its input, each of which represents the new, modified, or deleted zone map. In the case of a deletion, the opcode marks the zone map to be deleted in the input flist.

The PCM_OP_ZONEMAP_COMMIT_ZONEMAP output is an flist that links zone maps and products.

To retrieve zone map information from the database, use PCM_OP_ZONEMAP_GET_ZONEMAP. The default implementation enables you to retrieve zone maps in several ways:

- You can use it get all the zone maps by providing a zone map type POID
- You can use it get a specific zone map by providing the zone map name
- You can get the zone map associated with a product by providing the valid product POID

PCM_OP_ZONEMAP_GET_ZONEMAP is called by the Zone Mapper to retrieve zone map data and displays it in the user interface. It is also called by the zone map search opcode, PCM_OP_ZONEMAP_POL_GET_LINEAGE.

Customizing Zone Mapping

Use the following opcodes to customize zone mapping:

- PCM_OP_ZONEMAP_POL_GET_LINEAGE. See "Finding Zone Maps".
- PCM_OP_ZONEMAP_POL_GET_ZONEMAP. See "Getting Zone Maps from the BRM Database".
- PCM_OP_ZONEMAP_POL_SET_ZONEMAP. See "Saving Zone Map Data".
- PCM_OP_RATE_POL_EVENT_ZONEMAP. See "Getting Zone Maps and Impact Categories from the Pipeline Manager Database".

Finding Zone Maps

To search in a zone map for data associated with a given string, use PCM_OP_ZONEMAP_POL_GET_LINEAGE.

You supply a string and a zone map name. It then searches the zone map for the given string and returns the matching node with all ancestors of the matching node (the *lineage*).

This opcode calls the zone-based rating load opcode, PCM_OP_ZONEMAP_GET_ZONEMAP.

Getting Zone Maps from the BRM Database

To customize which zone map data to retrieve from the database, use PCM_OP_ZONEMAP_POL_GET_ZONEMAP. This opcode provides zone maps to display in the Zone Mapper.

This opcode retrieves the original zone map data from the BRM database, and depending on the calling program, it converts the binary format of zone map data into a readable flist, or it returns the binary format to be used in the lineage search.

The input flist must include either the PIN_FLD_POID for routing purposes or PIN_FLD_NAME to search by name.

Saving Zone Map Data

To customize how to save zone map data, use PCM_OP_ZONEMAP_POL_SET_ZONEMAP. This opcode saves zone map in the BRM database when you commit zone maps in the Zone Mapper.

This opcode retrieves the original zone map data from the BRM database, and depending on the calling program, it converts the binary format of zone map data into a readable flist, or it returns the binary format to be used in the lineage search.

The input flist must include either the PIN_FLD_POID for gateway routing purposes or PIN_FLD_NAME to search by name.

Getting Zone Maps and Impact Categories from the Pipeline Manager Database

To get zoning data from the Pipeline Manager database, use the PCM_OP_RATE_POL_EVENT_ZONEMAP policy opcode. This opcode returns the zone map name and impact category for an event from the Pipeline Manager database.

You can customize this policy to add new event classes that you have created, if those event classes use a real-time zoning pipeline.

PCM_OP_RATE_POL_EVENT_ZONEMAP performs the following functions:

1. Reads the event storable class.

2. Based on the service, it retrieves the calling number, the called number, and the zone map name from the event.
3. Prepares the input flist for a real-time zoning pipeline input module.
4. If it finds a matching impact category for the event zone, returns the impact category for the event.
5. If it doesn't find a matching impact category, returns an empty string and logs an error.

PCM_OP_RATE_POL_EVENT_ZONEMAP returns the zone map name and impact category for the event. If the opcode fails, it returns an empty string and logs an error.

Using the Rate Plan Selector in Pricing Center

Before you use the rate plan selector, you must create impact categories. See "Creating Impact Categories".

When you use a rate plan selector in Pricing Center, you start with an empty rate plan selector. After you provide a name for the rate plan selector, and (optionally) choose a zone map, you specify the attribute type and attributes you want to use.

To choose the attribute type, you click the appropriate option. To choose attributes, you select from a list of all attributes in the object.

To add rate plans and impact categories, you choose from lists. You can create a rate plan if it is not on the list, but impact categories must already be defined.

Set the impact category to **default** to apply the balance impacts of the rate plan according to standard rating attributes, and ignore all impact category conditions.

Using Wildcards to Match Event Data

In a rate plan selector, use an asterisk (*) as a wildcard as follows:

- In attribute columns, enter an asterisk to match any value in an event.
- In the **Impact Category** column, do not use an asterisk. BRM does not support its use as a wildcard in this column.

  See "Using Event Attributes to Define Impact Categories" for information about using wildcard characters in rate plan balance impacts.

Specifying the Priority for Attributes

The order of the rows in the rate plan selector determines the order in which rate plans are processed. The top row has the top priority. For example, you can specify to use one rate plan for one value of an attribute, and another rate plan for all other values. To do this, you give the value the top priority as shown in Figure 32-5:

Figure 32-5 Rate Plan Selector

Row	EVENT.PIN_FLD_CALL.PIN_FLD_TYPE	+...	Rate Plan	Impact Category
1	5		Calling card	default
2	*		Non-card	default

When you use multiple attributes, they are processed in the order from left to right.

33
Migrating Pricing Data from Legacy Databases

This chapter describes the process of migrating legacy pricing and discounting data to the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager database.

Topics in this document:

To migrate legacy data, you should be familiar with database administration tasks and must:

- Have in-depth knowledge of your legacy database.
- Have in-depth knowledge about the Pipeline Manager database.
- Know XML programming.
- Have a good understanding of XSD. See "Working with and Modifying the XSD".
- Understand the objects that are supported by the Pipeline Manager database and know the order in which the objects must be loaded into the Pipeline Manager database. For information, see "About the Types of Objects to Migrate".

> **Note:**
>
> Pipeline Configuration Manager is an optional component.

About Migrating Legacy Data

The Pipeline Configuration Manager is a set of scripts and utilities you can use to load data into the Pipeline Manager database.

Use the **LoadIfwConfig** utility to load pricing data into the Pipeline Manager database. You can migrate legacy data by adding the data to an XML file and running the **LoadIfwConfig** utility to load the data.

Any type of legacy database such as Oracle can be migrated to the Pipeline Manager database by using the **LoadIfwConfig** utility.

You use the **LoadIfwConfig** utility to migrate each table of your legacy database to the Pipeline Manager database. For more information, see "Loading Legacy Data into the Pipeline Manager Database with LoadIfwConfig".

During data migration, if the legacy system and Pipeline Manager are run in parallel, keep the databases synchronized by migrating data every time you make a change in the legacy database.

Overview of the Migration Process

The migration process follows these steps:

1. Planning the migration.

   This step includes comparing the data in the legacy database and the Pipeline Manager database and figuring out how to migrate the data from one database structure to another. See "Guidelines for Mapping Legacy Data".

2. Extracting legacy data to an XML file by using a data transformation program. See "About Migrating Legacy Data ".

3. Loading data into the Pipeline Manager database by using the **LoadIfwConfig** utility. See "Loading Legacy Data into the Pipeline Manager Database with LoadIfwConfig".

Legacy Data XML File Example

Legacy data representation in the XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<IFW xmlns ="http://www.portal.com/tools/pricing" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.portal.com/tools/pricing
IfwConfig.xsd">
<UOM UnitsOfMeasurement="hp1" Name="abc"/></IFW>
```

Guidelines for Mapping Legacy Data

You map the legacy pricing data to the pipeline pricing data in an XML file. This XML structure is defined in the XSD (**IfwConfig.xsd**). See "Working with and Modifying the XSD".

The structure of legacy pricing data is almost always different from the structure used by Pipeline Manager. For example, the legacy data might have pricing element equivalent to a pipeline charge, but not the equivalent of a pipeline charge version.

When you map legacy data, compare the structure of your legacy data against the structure of the Pipeline Manager data. Table 33-1 shows a sample comparison:

Table 33-1 Comparing Legacy and Pipeline Manager Data

Legacy Data	Pipeline Manager Data
Charge	Charge
No equivalent	Charge version
Charge configuration	Charge configuration
Charge model	*No equivalent*

In this example:

- You must create charge versions. This data might be in another part of the legacy data; for example, the version information might be included in the legacy charge itself.
- You must map the data in the legacy charge model to one of the pipeline charge elements.
- The legacy data and the Pipeline Manager data have charges and charge configuration. However, you must ensure that those elements include the same data.
- The XML file you create must use the structure defined in the XSD.

If you cannot directly map the data, then export the legacy data to flat files, and run a data transformation program on the flat files to produce data that can be loaded into the Pipeline Manager database. You can use any third-party tool for this purpose, depending on the type of legacy database. XML is a widely accepted programming language so many parsers and tools are available to transform data to XML.

Charge XML File Example

The following examples show the charge structure from the XSD file, and a charge in the XML file:

Charge definition in the XSD file

```
<xs:element name="RatePlan">
  <xs:annotation>
    <xs:documentation>...</xs:documentation>
  <xs:annotation>
  <xs:complexType>
    <xs:attribute name="RatePlanCode" type="xs:string" use="required"/>
    <xs:attribute name="Name" type="xs:string"/>
    <xs:attribute name="Status" default="D">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:length value="1"/>
          <xs:minLength value="1"/>
          <xs:maxLength value="1"/>
          <xs:enumeration value="S"/>
          <xs:enumeration value="T"/>
          <xs:enumeration value="A"/>
          <xs:enumeration value="D"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="CurrencyCode" type="xs:string" use="required" fixed="USD"/>
    <xs:attribute name="IsSplitting" type="xs:boolean" default="0"/>
    <xs:attribute name="Type" default="R">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:length value="1"/>
          <xs:minLength value="1"/>
          <xs:maxLength value="1"/>
          <xs:enumeration value="W"/>
          <xs:enumeration value="R"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="TaxTreatment" type="xs:boolean" use="required"/>
    <xs:attribute name="CalendarCode" type="xs:string" use="required"/>
    <xs:attribute name="UTCTimeOffset" type="xs:string" default="+0100"/>
```

```
  </xs:complexType>
</xs:element>
```

Charge representation in the XML file

```
<RatePlan
RatePlanCode="ratepc"
Name="rateplanc"
Status="D"
CurrencyCode="USD"
IsSplitting="0"
Type="R"
TaxTreatment="1"
CalendarCode="cal2001"
UTCTimeOffset="+0100"/>
```

Mapping Legacy Data

You extract the legacy data to a file by using a data conversion program. After the data is extracted from the legacy database, you migrate the data into an XML file. (You can also extract the data directly into an XML file.) You can insert different types of data, objects, and attributes in a single XML file.

To map legacy data, create an XML file that uses the structure defined in the XSD. Then migrate the objects in the specified order. See "About the Types of Objects to Migrate".

About the Types of Objects to Migrate

The objects that can be migrated to the Pipeline Manager database are defined in the XSD.

Table 33-2 shows the objects that can be migrated to the Pipeline Manager database and the prerequisite objects required for some objects. The prerequisite objects are the parent objects, which have objects dependent on them.

You must load child objects in the order shown in the table, as data in some objects depends on the existence of other objects. You load parent objects before child objects. However, sometimes a parent object can be loaded after loading a child object if the field in the child object that relates to parent object is not a mandatory field. For example, the Geomodel object is a parent of the Zonemodel object. However, the **Geomodel** field in the Zonemodel object is not a mandatory field, so you can load the Zonemodel object first.

Table 33-2 Order to Be Used in Loading Child Objects

Objects	Parent Object Prerequisites
Edrcdesc	None
Edrcfield	Edrcdesc
Aliasmap	Edrcfield Edrcdesc
Calendar	None
Special day	Calendar
Pipeline	Edrcdesc
Uom	None
Currency	None

Table 33-2 (Cont.) Order to Be Used in Loading Child Objects

Objects	Parent Object Prerequisites
Resource	Currency
Taxcode	None
Taxgroup	None
Tax	Taxgroup Taxcode
GLaccount	Taxcode
Revenuegroup	None
Rum	None
Rumgroup	None
Rumgrouplnk	Rumgroup Rum Uom
Service	GLaccount Revenuegroup Rumgroup
Serviceclass	Service
Refmap	None
Mapgroup	None
Servicemap	Mapgroup Service
Usageclass	None
Usageclassmap	Usageclass Mapgroup
Uscgroup	None
Rscgroup	None
Apngroup	None
Zonemodel	Geomodel Apngroup
Impactcat	None
Geomodel	Ruleset
Geoarealnk	Geomodel
Geozone	Zonemodel Service Impactcat
Standardzone	Service Zonemodel Impactcat
Usagetype	None

Table 33-2 (Cont.) Order to Be Used in Loading Child Objects

Objects	Parent Object Prerequisites
Uscmap	Usagetype Zonemodel Uscgroup Impactcat
Apnmap	Apngroup Impactcat
Discarding	Pipeline
Daycode	None
Timeinterval	None
Timezone	None
Timemodel	None
Timemodellnk	Timezone Daycode Timeinterval Timemodel
Pricemodel	None
Pricemdlstep	Pricemodel Resource Rum GLaccount Revenuegroup
Discountmodel	None
Dscmdlver	Discountmodel
Discountmaster	None
Discountdetail	Discountmaster
Dsctrigger	None
Dsccondition	Dsctrigger
Discountrule	Discountmaster
Discountstep	Discountrule
Dscmdlcnf	Discountrule Dsctrigger Dscmdlver
Rateplan	Calendar Currency
Rateplanver	Zonemodel Rateplan

Table 33-2 (Cont.) Order to Be Used in Loading Child Objects

Objects	Parent Object Prerequisites
Rateplancnf	Service Rateplanver Pricemodel Timezone Timemodel Impactcat
Rateadjust	Rateplanver
Specialdayrate	None
Specialdaylnk	Rateplanver Specialdayrate
Scenario	Edrcdesc
Uommap	Uom Rum
Exchangerate	Currency
Rscmap	Serviceclass Rscgroup
Rule	None
Ruleitem	None
Ruleset	None
Rulesetlist	Ruleset
Sla	Uscgroup Rscgroup Ruleset
Splittingtype	Pipeline
IcDaily	None
IcDailyalternate	None
Condition	Scenario Edrcfield
Icproduct	None
Networkoper	Taxgroup Currency
Networkmodel	Rateplan Networkoper Currency
Icproductgrp	Networkmodel
Icproductcnf	Icproduct Icproductgrp

Table 33-2 (Cont.) Order to Be Used in Loading Child Objects

Objects	Parent Object Prerequisites
Icproductrate	Icproduct Networkoper Rateplan Networkmodel
Noproduct	Networkoper Currency
Noproductcnf	ICproduct Timezone Noproduct Impactcat
Nosp	Mapgroup
Poi	None
Segment	None
Segratelnk	Rateplan Segment
Segzonelnk	Segment Zonemodel
Ciberocc	Networkoper
Socialnumber	None
Seqlogout	None
Dbversion	None
Iscript	None
Tam	None
Seqcheck	None
SeqlogIn	None
Duplicatecheck	None
Csstate	None
Destindesc	None
Classitem	None
Class	None
Queue	None
Changeset	None
Switch	Networkoper
Trunk	Switch Networkoper
Grouping	Scenario Edrcfield
Groupingcnf	Grouping Class

Table 33-2 (Cont.) Order to Be Used in Loading Child Objects

Objects	Parent Object Prerequisites
Classcon	GroupingCnf
Poiarealnk	Networkmodel Poi
Nobillrun	Networkoper Networkmodel
Aggregation	Scenario Edrcfield
Classlnk	Class Classitem
Classconlnk	Classitem Classcon
Dictionary	Queue
Trunkcnf	Trunk IcProductgrp Networkmodel Poi Switch Networkoper
Lergdata	Geoarealnk
Dscbalimpact	Discountstep
Cslock	Changeset
Csaudit	Changeset
Csreference	Changeset

Working with and Modifying the XSD

The XSD describes the structure of the XML document. The XML file you create must comply with the structure defined in the XSD.

The XSD defines the following items for an XML file:

- The elements and attributes, their data types, and the default and fixed values for the elements and attributes.
- Elements that are child elements, and the number and order of the child elements.
- If an element can be empty or can include text.

You can also modify the XSD to provide default values for certain attributes such as status, and provide domain constraints to the attributes. For example, the status can be A=active, D=de-active, T=test, or S=simulation. If the status S is not valid, you can restrict the XSD to allow only the status A, D, or T.

> **Note:**
>
> The structure of the default XSD cannot be changed.

Before Loading Legacy Data

Before you begin to load the legacy data, you must do the following:

- Install and configure the Pipeline Manager database.
- Prepare the legacy data for mapping. See "Guidelines for Mapping Legacy Data".
- Map the legacy data to Pipeline Manager database format. See "Mapping Legacy Data".

Configuring the Registry File for LoadIfwConfig Utility

The **LoadIfwConfig.reg** file provides database connection information to the **LoadIfwConfig** utility. You can edit this file manually, but it is also updated when you run the **pin_setup** utility.

The **LoadIfwConfig.reg** file is located in *pipeline_home***/tools/XMLloader.**

Most of the entries are standard connection entries, with these exceptions:

- Use the **LogFileName** entry to specify the file where debug messages are written.

> **Note:**
>
> To record debug messages, you must use the **verbose on** command when you run the **LoadIfwConfig** utility.

- Use the **LoadDataFromDB** entry to increase performance. Enabling this entry loads all the pricing data from the database into memory, where the utility can access it faster.

Sample **LoadIfwConfig.reg** file:

```
LOADIFWCONFIG
{
DataPool
   {
      Database
      {
         ModuleName = DbInterface
         Module
        {
            DatabaseName = dduttadb
            UserName     = PIN
            PassWord     = password
            AccessLib    = oci61
         }
      }
   }
XMLSchemaFile = Metadata.xml
LogFileName = LoadIfwConfig.log
LoadDataFromDB = False
}
```

Loading Legacy Data into the Pipeline Manager Database with LoadIfwConfig

To load legacy data into the Pipeline Manager database:

1. Go to *pipeline_home*/**bin**.
2. Run **LoadIfwConfig** using one of the following commands:
   - **Non-interactive mode**

     LoadIfwConfig [-i] *input_file* **[-I] [-c]**
   - **Interactive mode**

     LoadIfwConfig [read *input_file***] [Insert] [commit]**

> **Note:**
>
> The utility must be run from the directory where the XML file is located.

The **LoadIfwConfig** utility uploads the data from the XML file into the Pipeline Manager database and commits the data. If there is failure, the **LoadIfwConfig** utility rolls back the data and notifies you on your computer screen.

Deleting Data from the Pipeline Manager Database with LoadIfwConfig

To delete an object in a table, you must specify the primary field of the object in the input XML file. The object can be deleted only if the primary field has no dependent child objects.

To delete data from the Pipeline Manager database:

1. Go to *pipeline_home*/**bin**.
2. Run **LoadIfwConfig** using one of the following commands:
   - **Non-interactive mode**

     LoadIfwConfig [-i] *input_file* **[-p] [-c]** *output_file*
   - **Interactive mode**

     LoadIfwConfig [read *input_file***] [delete] [commit]** *output_file*

     Only the dependent objects are deleted, and the deleted objects are backed up in the output file.

Deleting Data Sample XML File

Deleting data representation in the XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<IFW xmlns="http://www.portal.com/tools/pricing" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.portal.com/tools/pricing
```

```
IfwConfig.xsd">
<UOM UnitsOfMeasurement="hp1" Name="abc"/></IFW>
```

Updating Data with the LoadIfwConfig Utility

To update data from the Pipeline Manager database:

1. Go to *pipeline_home*/**bin**.
2. Run **LoadIfwConfig** using one of the following commands:
   - **Non-interactive mode**

     LoadIfwConfig [-i] *input_file* **[-p]** *output_file*
   - **Interactive mode**

     LoadIfwConfig [read *input_file***] [update] [commit]** *output_file*

Exporting Data from the Database with the LoadIfwConfig Utility

To export data in a table, you must specify the primary field of the object in the input XML file. The utility exports all dependent data. For example, if you specify to export charges, the utility exports charge versions, charge configurations, pricings, impact categories, and time zones.

This sample shows an the contents of an XML file for exporting data:

```
<?xml version="1.0" encoding="UTF-8"?>
<IFW xmlns="http://www.portal.com/tools/pricing" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.portal.com/tools/pricing
IfwConfig.xsd">
<UOM UnitsOfMeasurement="hp1" Name="abc"/></IFW>
```

To export data from the Pipeline Manager database:

1. Go to *pipeline_home*/**bin**.
2. Run **LoadIfwConfig** using one of the following commands:
   - **Non-interactive mode**

     LoadIfwConfig [-i] *input_file* **[-f] [-r] [-o]** *output_file*
   - **Interactive mode**

     LoadIfwConfig [read *input_file***] [fetch] [write] [commit]** *output_file*

The exported objects are written in XML format.

Troubleshooting

Errors during data loading, deleting, and exporting are usually caused by the following:

- The XML data does not comply with the XSD structure.
- You are trying to load data into a table that does not exist.
- You are trying to load data but the prerequisite data does not exist. See "About the Types of Objects to Migrate".
- You are trying to delete data that has other data depending on it.

34

Improving Real-Time Rating Performance

This chapter provides information on evaluating and improving the performance of Oracle Communications Billing and Revenue Management (BRM) real-time rating.

Topics in this document:

Improving Real-Time Rating Performance

You can improve real-time rating performance by doing the following:

Changing the Precision of Rounded and Calculated Values

To improve performance, you can change the precision of rounded values and of values calculated by real-time rating. You change the precision by adding or modifying entries in the CM **pin.conf** file:

- To change the precision of rounded values, add or change the **rating_quantity_rounding_scale** entry. The value of this entry determines the number of digits to the right of the decimal place for rounding quantities. The default is **8**.
- To change the precision of calculated values, add or change the **rating_max_scale** entry. The value of this entry determines the number of digits to the right of the decimal place that are used. The default is **10**.

> **Note:**
>
> You must stop and restart the CM after you change these values.

Setting the Interval for Checking for Price List Changes

You can set the interval at which BRM checks for changes to the price list. If you change the price list frequently, you may want to use a shorter interval. If your price list is less volatile, you can increase the interval.

To change the interval:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Edit the following entry:

   ```
   - fm_rate refresh_product_interval 3600
   ```

   The value of this entry determines the interval in seconds. The default is **3600**.
3. Save the file.
4. Stop and restart the CM.

Setting the Interval for Updating Zone Maps

To specify how frequently BRM checks for changes to zone maps and updates them in the database:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Edit the following entry:

   ```
   - fm_zonemap_pol update_interval 3600
   ```

   The value of this entry determines the interval in seconds. The default is **3600**.
3. Save the file.
4. Stop and restart the CM.

Filtering the ERAs Considered during Rating and Discounting

By default, real-time rating checks for both account-level extended rating attributes (**/profile/acct_extrating** object) and service-level ERAs (**/profile/serv_extrating** object) when it searches for rating and discounting criteria. You can improve real-time rating performance by filtering the types of ERAs that BRM considers when it searches for rating and discounting criteria. For example, you can configure BRM to search for service-level ERAs only or to omit the ERA search altogether.

You can specify the types of ERAs to consider by modifying a field in the **rating** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To specify the ERA types:

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

   This command creates the XML file named **bus_params_rating.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.
2. Search the XML file for following line:

   ```
   <EnableEras>serviceAndAccount</EnableEras>
   ```
3. Change **serviceAndAccount** to one of the following:
   - **account**: Limits the rating and discounting criteria search to account-level ERAs by retrieving only the **/profile/acct_extrating** object.

- **service**: Limits the rating and discounting criteria search to service-level ERAs by retrieving only the **/profile/serv_extrating** object.
- **disabled**: Omits ERAs from the rating and discounting criteria. Because neither object is retrieved, this option provides the best performance.

> **Note:**
>
> BRM uses the XML in this file to overwrite the existing **rating** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM rating configuration.

4. Save the file.
5. Change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
6. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_rating.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
7. To verify that all fields are correct, you can display the **/config/business_params** object by using Object Browser in Developer Center or by using the **robj** command with the **testnap** utility.

   For more information, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*.
8. Stop and restart the CM.
9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Enabling and Disabling the Caching of Customized Products

When you use advanced customization to create customized products, BRM uses the customized products for rating. You can control whether customized products are cached for use by the real-time rating engine.

- If you choose not to cache customized products (the default setting), the real-time rating engine retrieves customized product data from the database during rating. This slows rating performance but minimizes the memory impact of customized products.
- If you choose to cache customized products, the CM size grows as customized products are created. Because the products are cached in memory, however, rating performance is increased.

You enable product caching by changing the **EnableTailormadeCache** field in the **rating** instance of the **/config/business_params** object from **0** to **1**. You can disable caching by changing the field back to **0**.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. See "pin_bus_params" in *BRM Developer's Guide*.

To enable caching of customized products:

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

   This command creates the XML file named **bus_params_rating.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:

   ```
   <EnableTailormadeCache>0</EnableTailormadeCache>
   ```

3. Change **0** to **1**.

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **rating** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM rating configuration.

4. Save the file.
5. Change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
6. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_rating.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

7. To verify that all fields are correct, you can display the **/config/business_params** object by using Object Browser or by using the **robj** command with the **testnap** utility.

   For information on using **testnap**, see "testnap" in *BRM Developer's Guide*.

8. Stop and restart the CM.
9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Configuring the Maximum Number of Products and Discounts Cached

Products and discounts that are used during the real-time rating process are automatically stored in the CM cache. This improves rating performance, but, over time, it can consume a large amount of memory. To prevent the CM cache from growing too large, you can set a maximum number of products and discounts that can be stored in the CM cache.

- When you set the maximum to a nonzero value, BRM prevents the real-time rating engine from storing more than the specified number of products and discounts in CM cache. When the maximum number is reached, BRM flushes 10% of the products and discounts from cache that have been used the least.

> **Note:**
>
> The maximum number of products and discounts that should be stored in CM cache depends on the your business needs.

- When you set the maximum to zero, BRM does not regulate the number of products and discounts stored in the CM cache. This is the default.

You configure the maximum number of products and discounts that can be cached by configuring the **ProductsDiscountsThreshold** field in the **rating** instance of the **/config/business_params** object.

To set a maximum number of products and discounts that can be cached:

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

   This command creates the XML file named **bus_params_rating.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:

   ```
   <ProductsDiscountsThreshold>0</ProductsDiscountsThreshold>
   ```

3. Change **0** to the maximum number of products and discounts that you would like stored in cache. The default value of **0** specifies to not regulate the number of products and discounts in the CM cache.

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **rating** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM rating configuration.

4. Save the file.
5. Change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
6. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_rating.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

7. To verify that all fields are correct, you can display the **/config/business_params** object by using Object Browser or by using the **robj** command with the **testnap** utility.

   For information on using **testnap**, see "testnap" in *BRM Developer's Guide*.

8. Stop and restart the CM.
9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Improving Performance for Loading Large Price Lists

If you have a large price list, you can improve performance in the following ways:

- Cache pricing data, such as G/L IDs and resource IDs. In a test environment where you modify your price list often, caching pricing data improves performance because there is no need to load price reference objects every time you commit the price list to the database.

  > **Note:**
  >
  > Pricing data is created every time the CM starts. Whenever the pricing data is changed in the database, the CM must be stopped and restarted to place the new information into the cache.

  In a production system where you rarely modify your price list, you do not need to cache pricing data. This reserves the CM cache for other uses and eliminates the need to stop and restart the CM to update the cache if you change the price list.

- Turn off event logging for price list creation events.

  > **Note:**
  >
  > When you turn off event logging, BRM still stores audit trail information for products, deals, and plans; however, there will not be an event log of when the price plans were modified and who modified them.

To improve loading performance:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Edit the **cache_references_at_start** entry. The default is **1** (cache data).

   ```
   - fm_price cache_references_at_start 1
   ```

3. Edit the **fm_price_prod_provisioning_cache** entry. The default entries are usually sufficient. For more information, see the instructions in the configuration (**pin.conf**) file.

   ```
   - cm_cache fm_price_prod_provisioning_cache 100, 102400, 13
   ```

4. Edit the **fm_price_cache_beid** entry. The default entries are usually sufficient. For more information, see the instructions in the configuration (**pin.conf**) file.

   ```
   - cm_cache fm_price_cache_beid 200, 524288, 32
   ```

5. Edit the **log_price_change_event** entry. The default is **0** (events are not logged).

   ```
   - fm_price log_price_change_event  0
   ```

6. Edit the **fm_offer_profile_cache** entry. The default entries are usually sufficient. For more information, see the instructions in the configuration (**pin.conf**) file.

   ```
   - cm_cache fm_offer_profile_cache 20, 102400, 13
   ```

> **Note:**
>
> For policy-driven charging, the **fm_offer_profile_cache** entry must be present in the **pin.conf** file. See "Policy-Driven Charging".

7. Save the file.
8. Stop and restart the CM.

35
Pricing Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) pricing utilities.

Topics in this document:

load_brm_pricing

Use this utility to load real-time and pipeline pricing data into the database. It runs the following utilities to load the pricing data:

- **loadpricelist** loads the real-time pricing data into the BRM database. See "loadpricelist" for more information.
- **LoadIfwConfig** loads the pipeline pricing data into the Pipeline Manager database. See "LoadIfwConfig" for more information.

Location

BRM_home/**bin**

Syntax

Load data:

```
load_brm_pricing [-v] [-d] [-f] -c Real_time_data_filename -F Pipeline_data_filename
```

Retrieve data:

```
load_brm_pricing [-v] [-d] [-f] -r Real_time_data_filename -F Pipeline_data_filename
                 [-P] [-D] [-S][-s ServiceType] [-t ModifiedTime] [-N name]
```

Get help:

```
load_brm_pricing -h
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands.

-d
Creates a log file for debugging purposes. Use this parameter for troubleshooting when the utility runs with no errors but the event map is not loaded into the database.

-f
Runs the command without prompting for a confirmation.

-c *Real_time_data_filename*
Reads the pricing data from *Real_time_data_filename* and commits it to the BRM database.

> **Note:**
>
> When you use the **-c** parameter, it must immediately precede the file name. The **-v**, **-d**, and **-f** parameters can be used in any order, but must precede **-c** *Real_time_data_filename*.

-F *Pipeline_data_filename*
Reads the pricing data from or writes the pricing data to *Pipeline_data_filename*.

- When used with the **-c** parameter, **-F** *Pipeline_data_filename* reads the pricing data from *Pipeline_data_filename* and commits it to the Pipeline Manager database.
- When used with the **-r** parameter, **-F** *Pipeline_data_filename* retrieves the pricing data from the Pipeline Manager database and writes it to *Pipeline_data_filename*.

-r *Real_time_data_filename* [-P] [-D] [-S] [-s *ServiceType*] [-t *ModifiedTime*] [-N *name*]
Retrieves pricing objects from the BRM database and writes the pricing data to *Real_time_data_filename*.

> **Note:**
>
> When you use the **-r** parameter, it must immediately precede the file name. The **-v**, **-d**, and **-f** parameters can be used in any order, but must precede **-r** and the file name. The parameters for determining the subset of information can be used in any order *after* the file name.

Use **-r** *Real_time_data_filename* with no additional parameters to retrieve all types of pricing objects from the BRM database.
Use the following parameters with **-r** *Real_time_data_filename* to retrieve only a subset of pricing objects from the BRM database:

- **-P** retrieves only the **/product** objects from the database.
- **-D** retrieves only the **/discount** objects from the database.
- **-S** retrieves only the **/sponsorship** objects from the database.
- **-s** *ServiceType* retrieves objects based on the specified service type. You can list multiple service types by using a comma (**,**) as a delimiter. For example: **-s /service/telco/gsm/telephony, /service/telco/gsm/data**.

> **Note:**
>
> Do not use **-s** *ServiceType* with the **-S** parameter.

- **-t** *ModifiedTime* retrieves objects that were modified after the specified timestamp. You specify time using the ISO-8601 standard. Table 35-1 lists the supported formats.

Format	Time Zone
YYYY	Local time of system used to run **load_brm_pricing**.
YYYY-MM	Local time of system used to run **load_brm_pricing**.
YYYY-MM-DD	Local time of system used to run **load_brm_pricing**.
*YYYY-MM-DD*T*hh:mm*Z	UTC
*YYYY-MM-DD*T*hh:mm:ss*Z	UTC
*YYYY-MM-DD*T*hh:mm*[+|-]*TZD*	*TZD* is the time zone relative to UTC. You can use a negative or positive offset from 00:00 to 12:00 (for example, **-05:00** or **+10:00**).
YYYY-MM-DDThh:mm:ss[+|-]*TZD*	*TZD* is the time zone relative to UTC. You can use a negative or positive offset from 00:00 to 12:00 (for example, **-05:00** or **+10:00**).

- **-N** *name* retrieves objects based on the specified charge offer, discount offer, or sponsorship.

> **Note:**
>
> You can use an asterisk (*) as a wildcard character to match zero or more characters in the name (for example, **-N** ***GSM***). Use **-N** with **-P**, **-D**, or **-S** to indicate the object type to be retrieved.

-h
Displays the syntax for the **load_brm_pricing** utility.

Results

The **load_brm_pricing** utility notifies you when it successfully completes a command.

If the utility does not notify you that it was successful, look in the utility log file to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

If an error occurs, enter the command again. The **load_brm_pricing** utility does not commit the pricing data to the database until it completes the command.

loadchangesets

Use this utility to import or export pipeline pricing data change sets. You can use the utility in interactive mode or non-interactive mode. In interactive mode, you enter single-word commands to perform individual actions.

You specify the BRM server to export from and the server to import to in a configuration file.

Location

*Pricing_Center_home***/lib**

Syntax: Non-Interactive Mode

```
loadchangesets    [-v] [-fi change_set_file]
                  [-v] [-fx change_set_file]
```

Parameters: Non-Interactive Mode

-v
Displays information about successful or failed processing as the utility runs.

-f *change_set_file*
Mandatory parameter that must precede **-i** and **-x**. Forces execution without prompting for confirmation.

-i *change_set_file*
When used with the **-f** parameter, imports change sets from *change_set_file* to the database. The file name must include the extension **.exp**.
When used alone, switches the utility to interactive mode. See "Syntax: Interactive Mode".

-x
When used with the **-f** parameter, exports the change sets from the database to *change_set_file*. Must be preceded with the **-f** parameter. The file name must include the extension **.exp**.

Syntax: Interactive Mode

```
loadchangesets -i

[read change_set_file]
[export]
[write change_set_file]
[import]
[help]
[quit]
[list]
```

Parameters: Interactive Mode

-i
Switches the utility to interactive mode. After entering interactive mode, commands are single words without the utility name.

export
Exports change sets from the database into memory.

write *change_set_file*
Writes the change sets stored in memory to *change_set_file*. Be sure to include the file name extension **.exp**.

read *change_set_file*
Reads change sets from *change_set_file* and stores them in memory. The file name must include the extension **.exp**.

import
Imports the change sets stored in memory to the database.

list
Lists the change sets currently stored in memory.

help
Displays help the syntax for the utility.

quit
Exits the utility.

Results: Export

Exported files are initially created in the *Pricing_Center_home***/export** directory. The status of the exported change sets changes to **Exported**. If all change sets are exported successfully, the file moves automatically into the **/export/done** directory.

If there are any errors during export, the entire transaction is rolled back. In addition, the status of the change sets is automatically reset from **Exported** to **In Progress**.

Results: Import

The import process generates a log file called **import.log.0**. The log file contains information about validation errors, change sets that have passed validation, and change sets that have been imported successfully.

If all change sets are imported without any errors, the transaction is committed. If there are any errors, the entire transaction is rolled back.

load_event_map

Use this utility to load the event maps into the BRM database. You define the event maps in the *BRM_home***/sys/data/pricing/example/pin_event_map** file.

For more information about event maps, see "Mapping Events and Services".

> **Note:**
>
> - The **load_event_map** utility overwrites the entire event map. If you are updating the event map, you cannot load new mappings only. You load the entire event map each time you run the **load_event_map** utility. The **load_event_map** utility does not restrict to load any particular event type with a service. However, if the service types and event pairs are duplicated, the **load_event_map** utility reports an error.
> - To connect to the BRM database, the **load_event_map** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_event_map [-d] [-v] pin_event_map_file
```

Parameters

-d
Logs messages to the **default.pinlog** file in the current directory, or in a directory specified in the utility configuration file. Use this parameter for troubleshooting when the utility runs with no errors, but the event map is not loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_event_map** *other_parameter* **–v >** *filename***.log**

pin_event_map_file
The name and location of the file that defines event maps. The default is *BRM_home***/sys/data/pricing/example/pin_event_map**.
If you run the command from a different directory from the one in which *pin_event_map_file* is located, you must include the entire path for the file.

Results

The **load_event_map** utility notifies you only if it encounters errors. Look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started, or in a directory specified in the utility configuration file.

load_pin_beid

Use this utility to load resources into the BRM database. You define the resources in the *BRM_home***/sys/data/pricing/example/pin_beid** file.

> **Note:**
>
> It is easier to use the Resource Editor to define resources.

For more information about defining resources, see "About Resources".

> **Note:**
>
> - The **load_pin_beid** utility overwrites the existing resources. If you are updating resources, you cannot load new resources only. You load complete sets of resources each time you run the **load_pin_beid** utility.
> - When you add or edit resources by using the Resource Editor, the Resource Editor does not update the **pin_beid** file. Therefore, if you want the **pin_beid** file to include all of your resources, you must edit it by hand.
> - To connect to the BRM database, the **load_pin_beid** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_beid [-d] [-v] pin_beid_file
```

Parameters

-d
Sends all flists used in creating the resources to a log file. Use this parameter for debugging when the utility appears to have run with no errors, but the resources do not appear in the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_beid** *other_parameter* **–v >** *filename***.log**

pin_beid_file
The name and location of the file that defines resources. The default is *BRM_home***/sys/data/pricing/example/pin_beid**.
If you copy the **pin_beid** file to the same directory from which you run the **load_pin_beid** utility, you do not have to specify either the path or the file name.

Results

The **load_pin_beid** utility notifies you only if it encounters errors.

> **Note:**
>
> You must restart the Connection Manager to make the new resources available.

load_pin_impact_category

Use this utility to load impact categories into the BRM database. You define the impact categories for the rate plan selector charges in the *BRM_home***/sys/data/config/pin_impact_category** file.

For information about impact categories, see "About Impact Categories".

> **Note:**
>
> - The **load_pin_impact_category** overwrites the existing impact categories. If you are updating a set of impact categories, you cannot load new impact categories only. You load complete sets of impact categories each time you run the **load_pin_impact_category** utility.
> - To connect to the BRM database, the **load_pin_impact_category** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

load_pin_impact_category *pin_impact_category_file*

Parameters

pin_impact_category_file
The name and location of the file that defines the impact categories. The default is *BRM_home*/**sys/data/config/pin_impact_category**.
If you copy the **pin_impact_category** file to the same directory from which you run the **load_pin_impact_category** utility, you do not have to specify either the path or the file name.

Results

The **load_pin_impact_category** utility notifies you when it successfully creates the **/config/impact_category** storable object.

If the **load_pin_impact_category** utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

> **Note:**
>
> You must restart the Connection Manager to make the new impact categories available.

load_pin_rum

Use this utility to load ratable usage metrics (RUMs) into the **/config/rum** storable object in the BRM database. You define RUMs in the *BRM_home*/**sys/data/pricing/example/pin_rum** file.

For information about RUMs, see "Real-Time Rating Based on Multiple RUMs".

> **Note:**
>
> - The **load_pin_rum** utility overwrites the existing RUMs. If you are updating a set of RUMs, you cannot load new RUMs only. You load complete sets of RUMs each time you run the **load_pin_rum** utility.
> - To connect to the BRM database, the **load_pin_rum** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

BRM_home/**bin**

Syntax

```
load_pin_rum [-d] [-v] pin_rum_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the RUMs do not appear in the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_rum** *other_parameter* **–v >** *filename***.log**

pin_rum_file
The name and location of the file that defines RUMs. The default is *BRM_home***/sys/data/pricing/example/pin_rum**.
If you copy the **pin_rum** file to the same directory from which you run the **load_pin_rum** utility, you do not have to specify either the path or the file name.

Results

The **load_pin_rum** utility notifies you only if it encounters errors.

> **Note:**
>
> You must restart the Connection Manager to make the new RUMs available.

load_pin_spec_rates

Use this utility to set up customized rating by loading the contents of the **pin_spec_rates** file into the BRM database.

For information about customizing rating, see *BRM Opcode Guide*.

> **Note:**
>
> - The **load_pin_spec_rates** overwrites the existing setup for administrative events charges. If you are updating a set of administrative events charges, you cannot load new charges only. You load complete sets of charges each time you run the **load_pin_spec_rates** utility.
> - To connect to the BRM database, the **load_pin_spec_rates** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_spec_rates pin_spec_rates_file
```

Parameters

pin_spec_rates_file
The name and location of the file that maps opcodes to event types to be rated. The default is *BRM_home***/sys/data/config/pin_spec_rates**.
If you copy the **pin_spec_rates** file to the same directory from which you run the **load_pin_spec_rates** utility, you do not have to specify either the path or the file name.

Results

If the **load_pin_spec_rates** utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

> **Note:**
>
> You must restart the Connection Manager for the new administrative event charges to take effect.

load_pin_sub_bal_contributor

Use this utility to load the configuration for contributor-based sub-balances into the BRM database. You define the sub-balance configuration in the *BRM_home***/sys/data/pricing/example/pin_sub_bal_contributor** file.

For information about contributor-based sub-balances, see "Managing Sub-Balances".

> **Note:**
>
> - The **load_pin_sub_bal_contributor** utility overwrites the existing sub-balance configurations. If you are updating a set of sub-balance configurations, you cannot load new configurations only. You load complete sets of sub-balance configurations each time you run the **load_pin_sub_bal_contributor** utility.
> - To connect to the BRM database, the **load_pin_sub_bal_contributor** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.
> - The resources specified in the **pin_sub_bal_contributor** file must exist in the BRM database before running the **load_pin_sub_bal_contributor** utility. This includes any aggregation counters used to track consumption of resources that are shared among several accounts. Therefore, you must first load the **pin_beid** file by running the **load_pin_beid** utility. See "load_pin_beid".

Location

*BRM_home***/bin**

Syntax

```
load_pin_sub_bal_contributor [-d] [-v] pin_sub_bal_contributor_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the sub-balance configurations have not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_sub_bal_contributor** *other_parameter* **–v >** *filename***.log**

pin_sub_bal_contributor_file
The name and location of the file that defines the configuration for contributor-based sub-balances. The default **pin_sub_bal_contributor** file is in *BRM_home***/sys/data/pricing/sample**.
If you copy the **pin_sub_bal_contributor** file to the directory from which you run the **load_pin_sub_bal_contributor** utility, you do not have to specify either the path or the file name.

Results

The **load_pin_sub_bal_contributor** utility notifies you when it successfully creates the **/config/sub_bal_contributor** object.

If the **load_pin_sub_bal_contributor** utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This log file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

> **Note:**
>
> You must restart the Connection Manager to make the new sub-balance configurations available.

loadpricelist

Use this utility to prepare and commit price list information, such as products, deals, plans, offer profiles, and plan lists, to your BRM database. You define the price list or offer profile configurations by using the **price_list.xsd** file in the *BRM_home***/PricingCenter** directory.

The **loadpricelist** utility is the XML Pricing Interface.

> **Note:**
>
> - Do not use the **loadpricelist** utility to assign custom rateable usage metrics (RUMs) to fold events in any rate plans. Fold events cannot use custom RUMs; therefore, products configured with them are rated incorrectly.
> - To connect to the BRM database, the CLASSPATH should point to an **Infranet.properties** file with the correct login information for the **loadpricelist** utility. The default **Infranet.properties** file is in the **/common** directory. To put the **Infranet.properties** file in the local directory with the **loadpricelist** utility, add period-slash (**./**) to the beginning of the CLASSPATH.

For information, see "Using the XML Pricing Interface to Create a Price List ".

Location

*BRM_home***/bin**

Syntax: Non-Interactive Mode

Copy:

```
loadpricelist [-v] [-d] [-f] -c PriceListFile
```

Purge:

```
loadpricelist [-v] [-d] [-f] -p
```

Retrieve:

```
loadpricelist [-v] [-d] [-f] -r PriceListFile [-P] [-D] [-S] [-s ServiceType] [-t
ModifiedTime]
```

Get help:

```
loadpricelist -h
```

Parameters: Non-Interactive Mode

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands.

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-f
Runs the command without prompting for a confirmation.

-c *PriceListFile*
Reads the price list information from *PriceListFile* and commits it to the BRM database.

> **Note:**
>
> - When you use the **-c** parameter, it must immediately precede the file name. The **-v**, **-d**, and **-f** parameters can be used in any order, but must precede **-c** and the file name.
> - *PriceListFile* must be in the same directory as the **price_list.xsd** file. If they are not in the same directory, edit the **noNamespaceSchemaLocation** entry in *PriceListFile* to include the full path to **price_list.xsd**. By default, **price_list.xsd** is in the *BRM_home***PricingCenter** directory. For example, if the XSD file is located in **C:\pricelists**, change the entry to:
>
>   ```
>   xsi:noNamespaceSchemaLocation="C:\pricelists\price_list.xsd"
>   ```

-p
Purges pricing objects from the BRM database.

-r *PriceListFile* [-P] [-D] [-S] [-s *ServiceType*] [-t *ModifiedTime*]
Retrieves pricing objects from the BRM database and writes them to *PriceListFile*. When the **infranet.batchsize** entry is present in the **loadpricelist Infranet.properties** file, the utility retrieves objects in batches based on the value of the entry. See "Downloading Price Lists in Batches".
Use **-r** *PriceListFile* with no additional parameters to retrieve all types of pricing objects from the BRM database.
Use the following parameters with **-r** *PriceListFile* to retrieve only a subset of pricing objects from the BRM database:

- **-P** retrieves only the **/product** objects from the database.
- **-D** retrieves only the **/discount** objects from the database.
- **-S** retrieves only the **/sponsorship** objects from the database.
- **-s** *ServiceType* retrieves objects based on the specified service type. You can list multiple service types by using a comma (**,**) as a delimiter. For example: **-s /service/telco/gsm/telephony, /service/telco/gsm/data**.

> **Note:**
>
> Do not use **-s** *ServiceType* with the **-S** parameter.

- **-t** *ModifiedTime* retrieves objects that were modified after the specified timestamp. You specify time using the ISO-8601 standard. Table 35-2 lists the supported formats:

Format	Time Zone
YYYY	Local time of system used to run **loadpricelist**.
YYYY-MM	Local time of system used to run **loadpricelist**.
YYYY-MM-DD	Local time of system used to run **loadpricelist**.
*YYYY-MM-DD*T*hh*:*mm*Z	UTC
*YYYY-MM-DD*T*hh*:*mm*:*ss*Z	UTC
*YYYY-MM-DD*T*hh*:*mm*[+|-]*TZD*	*TZD* is the time zone relative to UTC. You can use a negative or positive offset from 00:00 to 12:00 (for example, **-05:00** or **+10:00**).
*YYYY-MM-DD*T*hh:mm:ss*[+|-]*TZD*	*TZD* is the time zone relative to UTC. You can use a negative or positive offset from 00:00 to 12:00 (for example, **-05:00** or **+10:00**).

> **Note:**
>
> When you use the **-r** parameter, it must immediately precede the file name. The **-v**, **-d**, and **-f** parameters can be used in any order, but must precede **-r** and the file name. The parameters for determining the subset of information can be used in any order *after* the file name.

-h
Displays the syntax for the **loadpricelist** utility.

Syntax: Interactive Mode

```
loadpricelist    read PriceListFile

                 write
                   PriceListFile [planlist [type] | plan |
                   | bestpricing | deal | product | discount |
```

```
  | sponsorship | offer_profile] [PriceObjectName]

write
  PriceListFile [dependency | transition]

delete
  [planlist [type] | plan | bestpricing | deal |
  | product | discount | sponsorship |
      offer_profile] [PriceObjectName]

delete
  [dependency | transition]

retrieve
  [-s ServiceType] [-t ModifiedTime] [product]
  [discount] [sponsorship]

commit
purge
help
quit
verbose [on | off | debug]
list
```

Parameters: Interactive Mode

read *PriceListFile*
Reads the price list information from *PriceListFile* and stores it in internal memory as an flist.

> **Note:**
>
> - **loadpricelist** maintains only one flist. When you load the price list information from your database, **loadpricelist** removes the current flist and creates a new flist.
> - *PriceListFile* must be in the same directory as the **price_list.xsd** file. If they are not in the same directory, edit the **noNamespaceSchemaLocation** entry in *PriceListFile* to include the full path to **price_list.xsd**. By default, **price_list.xsd** is in the *BRM_home***PricingCenter** directory. For example, if the XSD file is located in **C:\pricelists**, change the entry to: **xsi:noNamespaceSchemaLocation="C:\pricelists\price_list.xsd"**

write
Writes the price list information from the in-memory internal flist into *PriceListFile*.

```
write PriceListFile [planlist [type] | plan | bestpricing | deal | product | discount |
sponsorship | offer_profile] [PriceObjectName]

write PriceListFile [dependency | transition]
```

Use **write** *PriceListFile* without any other parameters to write *all* pricing objects from the in-memory internal flist into *PriceListFile*.
Use the following parameters to write only a subset of pricing objects from the in-memory internal flist into *PriceListFile*:

- **planlist** [*type*] specifies to write only plan list information into *PriceListFile*. You can optionally specify to write only a plan list object of a particular type into *PriceListFile*.
- **plan** specifies to write only the **/plan** objects into *PriceListFile*.
- **bestpricing** specifies to write only the **/bestpricing** objects into *PriceListFile*.
- **deal** specifies to write only the **/deal** objects into *PriceListFile*.
- **product** specifies to write only the **/product** objects into *PriceListFile*.
- **discount** specifies to write only the **/discount** objects into *PriceListFile*.
- **sponsorship** specifies to write only the **/sponsorship** objects into *PriceListFile*.
- **offer_profile** specifies to write only the **/offer_profile** objects into *PriceListFile*.
- *PriceObject* specifies to write only the specified pricing object into *PriceListFile*.
- **dependency** specifies to write only **/dependency** objects into *PriceListFile*.
- **transition** specifies to write only **/transition information** objects into *PriceListFile*.

delete
Deletes in-memory price list information from the BRM database.

```
delete [planlist [type] | plan | bestpricing | deal | product | discount | sponsorship
| offer_profile]
       [PriceObject]

delete [dependency | transition]
```

Use **delete** without any other parameters to delete *all* in-memory pricing information from the database.
Use the following parameters to delete only a subset of in-memory pricing information from the database:

- **planlist** [*type*] specifies to delete the plan list information of a particular type.
- **plan** specifies to delete only the **/plan** objects from the database.
- **bestpricing** specifies to delete only the **/bestpricing** objects from the database.
- **deal** specifies to delete only the **/deal** objects from the database.
- **product** specifies to delete only the **/product** objects and all related product information, such as the rate plan, rate, rate tier, and balance impact, from the database.
- **discount** specifies to delete only the **/discount** objects from the database.
- **sponsorship** specifies to delete only the **/sponsorship** objects from the database.
- **offer_profile** specifies to delete only the **/offer_profile** objects from the database.
- *PriceObject* specifies to delete only the specified pricing object from the database.
- **dependency** specifies to delete only **/dependency** objects.
- **transition** specifies to delete only **/transition** objects.

retrieve
Retrieves the specified price list objects from the BRM database and stores it in internal memory as an flist.

```
retrieve  [-s ServiceType] [-t ModifiedTime] [product] [discount] [sponsorship]
```

Use **retrieve** without any other parameters to retrieve *all* pricing objects from the BRM database.

Use the following parameters to retrieve only a subset of pricing objects from the BRM database:

- **-s** *ServiceType* retrieves objects based on the specified service type. Multiple service types are delimited with a comma (**,**).

  Do not use this parameter with the **sponsorship** parameter.

- **-t** *ModifiedTime* retrieves objects that were modified after the specified time. You specify time using the ISO-8601 standard. Table 35-3 lists the supported formats.

Format	Time Zone
YYYY	Local time of system used to run **loadpricelist**.
YYYY-MM	Local time of system used to run **loadpricelist**.
YYYY-MM-DD	Local time of system used to run **loadpricelist**.
*YYYY-MM-DD*T*hh:mm*Z	UTC
*YYYY-MM-DD*T*hh:mm:ss*Z	UTC
*YYYY-MM-DD*T*hh:mm*[+|-]*TZD*	*TZD* is the time zone relative to UTC. You can use a negative or positive offset from 00:00 to 12:00 (for example, **-05:00** or **+10:00**).
*YYYY-MM-DD*T*hh:mm:ss*[+|-]*TZD*	*TZD* is the time zone relative to UTC. You can use a negative or positive offset from 00:00 to 12:00 (for example, **-05:00** or **+10:00**).

- **product** retrieves only the **/product** objects from the database.
- **discount** retrieves only the **/discount** objects from the database.
- **sponsorship** retrieves only the **/sponsorship** objects from the database.

commit
Commits the pricing information to the database.

purge
Removes all price objects from the database.

help
Displays the syntax for the **loadpricelist** utility.

quit
Exits the utility.

verbose on | off | debug
Sets verbose information:

- **verbose on** displays the status of the command most recently run.
- **verbose off** displays the status only if there is an error.
- **verbose debug** logs the status of the command to the log file specified in the **Infranet.properties** file. The default log file is **javapcm.log**.

list
Prints to the screen all price list information currently stored in internal memory. Because the in-memory price list is in flist format, the utility displays the price list in flist format.

Results

The **loadpricelist** utility notifies you when it successfully completes a command.

If the utility does not notify you that it was successful, look in the **javapcm.log** file to find any errors. This log file is either in the directory from which the utility was started or in a directory specified in the Java property file.

If an error occurs, enter the command again. The **loadpricelist** utility does not save changes until it completes the command.

load_usage_map

Use this utility to load the usage map into the **/config/usage_map/system** storable object in the BRM database. You define the usage map in the *BRM_home*/**sys/data/pricing/example/pin_usage_map** file.

For more information, see "Mapping Event Types to RUMs".

> **Note:**
>
> - The **load_usage_map** overwrites the existing usage maps. If you are updating a set of usage maps, you cannot load new usage maps only. You load complete sets of usage maps each time you run the **load_usage_map** utility.
> - To connect to the BRM database, the **load_usage_map** utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

BRM_home/**bin**

Syntax

```
load_usage_map [-d] [-v] pin_usage_map_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the usage maps do not appear in the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_usage_map** *other_parameter* **–v >** *filename***.log**

pin_usage_map_file
The name and location of the file that defines usage maps. The default is *BRM_home***/sys/data/pricing/example/pin_usage_map**.
If you copy the **pin_usage_map** file to the same directory from which you run the **load_usage_map** utility, you do not have to specify either the path or the file name.

Results

The **load_usage_map** utility notifies you only if it encounters errors.

pin_apply_promotion

Use this utility to apply promotions, such as free data or minutes, to accounts based on special dates. This utility:

- Searches through all **/profile/specialdates** objects for a PIN_FLD_NEXT_DATE that is less than or equal to the current date and then applies the promotion.
- Sets the PIN_FLD_NEXT_DATE field in the **/profile/specialdates** object to the next occurrence of the special date.

For more information, see "Working with Promotions".

> **Note:**
>
> - To access the BRM database, this utility requires a configuration file (*BRM_home***/apps/pin_billd/pin.conf**) in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.
> - You can tune the performance of this multithreaded application (MTA) utility by using the MTA configuration file entries. See "Improving Billing Performance" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
pin_apply_promotion [-verbose] [-help]
```

Parameters

-verbose
Displays information about successful or failed processing as the utility runs.

-help
Displays the syntax for the **pin_apply_promotion** utility.

Results

The **pin_apply_promotion** utility writes the results to a log file, which is located in the path you specified in the utility's **pin.conf** configuration file.

pin_clean_offer_override

Use this utility to remove expired date range and pricing tag combinations from the **/offering_override_values** object.

For more information, see "Dynamically Changing One-Time and Recurring Fees Based On the Date".

> **Note:**
>
> - To access the BRM database, this utility requires a configuration file (*BRM_home***/apps/pin_subscription/pin.conf**) in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.
> - You can tune the performance of this multithreaded application (MTA) utility by using the MTA configuration file entries. See "Improving Billing Performance" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
pin_clean_offer_override [-end date] [-event eventType] [-verbose] [-help]
```

Parameters

-end *date*
Removes all date range and pricing tag combinations that expired prior to the specified date. The date must be in the format *MM**/**DD**/**YYYY*. If the date is not specified, the utility uses the current date.

-event *eventType*
Removes all date range and pricing tag combinations associated with the specified event type.

-verbose
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **pin_clean_offer_override** *other_parameter* **–v >** *filename***.log**

-help
Displays the syntax for the utility.

Results

The **pin_clean_offer_override** utility notifies you only if it encounters errors. Look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

Part III

Configuring Pipeline Discounting

This part describes how to configure pipeline discounting in Oracle Communications Billing and Revenue Management (BRM). It includes these chapters:

- Configuring Discounting Modules and Components
- Discounting Utilities

36

Configuring Discounting Modules and Components

This chapter describes how to configure discounting in the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Configuring a Batch Discounting Pipeline

Batch discounting is typically performed in a separate discounting pipeline.

Configure the following data modules:

- **DAT_AccountBatch**. This module provides account data to Pipeline Manager. This module loads account data into memory when you start Pipeline Manager, and updates it when it is changed in the BRM database.
- **DAT_BalanceBatch**. This module provides balance data for the FCT_Discount module when discounting is run in batch. This module loads balance data into memory when you start Pipeline Manager, and keeps the balance data synchronized between the Pipeline Manager database and the BRM database. See "DAT_BalanceBatch".
- **DAT_ModelSelector**. This module provides discount selector data to the FCT_DiscountAnalysis module. See "DAT_ModelSelector".
- **DAT_Discount**. This module supplies discount information to the discount function modules. See "DAT_Discount".

Configure the following function modules:

- **FCT_Discount**. This module performs the discount calculations and adds discounting data to the event data record (EDR). See "FCT_Discount".
- **FCT_DiscountAnalysis**. This module selects applicable discounts and prioritizes them. See "FCT_DiscountAnalysis".
- **FCT_Rounding**. This module rounds the balance impacts of discounting. Use the **Mode** registry entry to specify **Discounting**. See "FCT_Rounding".
- **FCT_ApplyBalance**. This module is used only for batch discounting. This module adds the discount balance impact to the EDR and updates the Pipeline Manager memory. See "FCT_ApplyBalance".

About Setting the Validity of Balance Elements Impacted by Discounts

The effective period of a granted balance element can start when a subscriber first consumes the balance element balance.

The following modules are used to set the validity period of balance elements that start on first usage when they are impacted for the first time:

- **DAT_BalanceBatch**. This module calculates the balance element validity period based on the EDR timestamp and initializes the validity period in memory.

If the validity periods of all first-usage balance elements in the bundle should be synchronized, DAT_BalanceBatch adds information about those balance elements to the EDR.

- **FCT_ApplyBalance**. This module sets the validity period information in the EDR for all first-usage balance elements whose validity needs to be set. It then sends the entire EDR to an output stream.

  You specify the first-usage validity output stream in the FCT_ApplyBalance registry. See "FCT_ApplyBalance".

  You configure the output stream in the batch rating pipeline. See "Configuring Pipeline Output for First-Usage Products, Discounts, and Balance Elements".

To set the validity period of first-usage balance elements sent to the output stream, you configure Universal Event (UE) Loader. See "About Updating Validity Period Information in the BRM Database".

Configuring Batch Discounting to Restrict Balance Element Validity End Time

When the validity period of a granted balance element starts on first usage and ends relative to the start time, you can restrict the balance element end time to ensure the balance element balance cannot continue to be consumed after the charge offer or discount offer expires.

To restrict the validity end time of first-usage balance elements, configure "DAT_BalanceBatch" to use the **RestrictResourceValidityToOffer** business parameter setting from the BRM database.

You configure DAT_BalanceBatch to use business parameter settings from the BRM database by performing the following:

- Configuring the "DAT_PortalConfig" module in your registry file. This module must be listed before all other data modules in the registry file.
- Connecting the "DAT_BalanceBatch" module to DAT_PortalConfig by using the PortalConfigDataModule registry entry.

When you restrict balance element validity end time, DAT_BalanceBatch sets the end time of the balance element validity period to the end time of the charge offer or discount offer that grants the balance element if it is earlier than the balance element validity end time.

Calculating the Match Factor of Parallel and Sequential Discounts

The *match factor* in discounting is the percentage of usage that is discounted by a single discount when more than one discount is applied. The match factor is used with cascading discounts in which a discount can be applied only to the portion of usage that has not already been discounted.

For example, if an account owns two cascading discounts and the first one discounts 75% of the usage, the match factor is .75. The second discount, therefore, can be applied only to 25% of the usage.

By default, discounting does not calculate the match factor for parallel and sequential discounts. Should you need to calculate the match factor for parallel and sequential discounts, set the **AvoidMatchFactorCalculation** entry to **False** in the FCT_Discount module registry. See "FCT_Discount".

Configuring a Real-Time Discounting Pipeline

You can use a real-time discounting pipeline to calculate discounts for events that are rated by real-time rating. This allows you to discount all events in real time, so customer discount balances are always current.

To configure real-time discounting:

1. Configure a real-time discounting pipeline. See "Configuring a Real-Time Discounting Pipeline".
2. Configure the Input registry section. See "Configuring the Input Registry Section".
3. Configure the Connection Manager (CM) to send discounting requests to the NET_EM module.

Configuring a Real-Time Discounting Pipeline

Configure a real-time discounting pipeline that includes the following real-time discounting modules:

- **INP_Realtime**. This module handles flist-to-EDR format translation for a real-time pipeline. See "INP_Realtime".

  Use this entry for the **OpcodeMapping** entry:

  ```
  OpcodeMapping = ./formatDesc/Formats/Realtime/discount_event.xml
  ```

- **OUT_Realtime**. This module handles EDR-to-flist format translation for a real-time pipeline. See "OUT_Realtime".
- **NET_EM**. This module provides an interface to the CM for the INP_Realtime and OUT_Realtime modules and an interface to the BRM database for the DAT_AccountRealtime and DAT_BalanceRealtime modules.
- **DAT_AccountRealtime**. This module provides account cycle data to the FCT_Discount module. The DAT_AccountRealtime module gets data from the BRM database by connecting with the NET_EM module. It does not store data in memory, so it does not load data when you start Pipeline Manager. See "DAT_AccountRealtime".
- **DAT_BalanceRealtime**. This module provides balance data for the FCT_Discount module for real-time discounting. The DAT_BalanceRealtime module gets data from the BRM database by connecting with the NET_EM module. It does not store data in memory, so it does not load data when you start Pipeline Manager. See "DAT_BalanceRealtime".
- **FCT_CreditLimitCheck**. This module is used only for real-time discounting. This module is used during the prepaid authorization process to determine whether event owners have enough balance elements in their account balance to use a requested service. For more information, see "FCT_CreditLimitCheck".

In addition, configure the following standard discounting modules:

- **FCT_Discount**. This module performs the discount calculations and adds discounting data to the EDR. See "FCT_Discount".
- **FCT_DiscountAnalysis**. This module selects applicable discounts and prioritizes them. See "FCT_DiscountAnalysis".

Configuring the Input Registry Section

To manage real-time discounting efficiently, you must enable the Input registry **UnitsPerTransaction** entry. Use the following entry in the Input registry section:

```
Input
{
   UnitsPerTransaction = 1
   NoThread = true
...
```

About Dumping Discount Information during Run Time

You can dump discount configuration information during run time. Dumping the information writes it to a file or to the terminal. This is useful should you need to verify, compare, or provide discount configuration information during run time for troubleshooting purposes.

You can use the DAT_Discount and DAT_ModelSelector modules to write discount configuration information. You can write discount configuration information for one or all discounts in your system by using the **DiscountModel** and **DataFileName** semaphores. See "DAT_Discount" and "DAT_ModelSelector".

About Discount Transaction Management

To maintain data integrity in a pipeline, discounting uses transactional processing on two levels:

- Standard pipeline transactions, managed by the Transaction Manager (TAM). If a transaction fails, the input is stopped and all open transactions are rolled back. After rolling back the data, the input is restarted.

  > **Note:**
  >
  > To enable transaction management, the **redoEnable** registry parameter of the TAM must be set to **True**. If the redo mechanism is not enabled, discounting will block any other sequential transactions.

- EDR transactions, managed by the discounting modules. An EDR might contain multiple charge packets that are manipulated by the FCT_Discount module. The module might find errors in some charge packets and not be able to finish processing the EDR. Therefore, all changes made in one EDR are logged. If there are no errors, the data is committed, otherwise the changes made by the module are rolled back and the EDR is not committed.

About Processing Balance Groups Locked by Other Transactions

During discount calculations in BRM, when discounting data is added to the event data record (EDR), the associated balance group is locked by the transaction. By default, there is a dependency between concurrent transactions involving the same balance group. If transaction A locks a balance group, then, by default, transaction B waits for transaction A to commit or roll back before it locks the same balance group.

You can enhance pipeline throughput performance in BRM by configuring the **IgnoreEDROnLock** entry in the FCT_Discount module to ignore an event data record, if the associated balance group object is locked by another transaction.

For more information on the **IgnoreEDROnLock** registry entry in the FCT_Discount module, see "FCT_Discount".

To configure the FCT_Discount module for:

- Batch operations, set the **IgnoreEDROnLock** registry entry to **True**.
- Run-time operations, set the **IgnoreEDROnLock** semaphore entry to **True**.

When the FCT_Discount module processes concurrent transactions involving the same locked balance group objects, if **IgnoreEDROnLock** entry is set to **True**, it places the ignored or rejected EDRs with the locked balance group in the **discountError** directory.

Configuring Custom Business Events for Pipeline Discounting

A business event is a BRM operation that you define in the Payload Generator EM configuration file (**payloadconfig.xml**).

To enable custom business events in Pipeline Manager, you need to list them in the DAT_BalanceBatch **CustomEvents** registry entry. For example, this entry enables an event named CycleRollover20days:

```
CustomEvents
{
   CycleRollover20days
 }
```

37
Discounting Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager discounting utilities.

pin_discount_cleanup

Use this utility to change the status of expired discounts from **active** to **canceled** and to delete canceled discounts.

You use this utility to close or delete the discounts that are canceled in the middle of a cycle and the discount's validity rule is set to Full discount.

You can run this utility daily or add it to the **pin_bill_day** script to be run automatically.

Location

BRM_home/**bin**

Syntax

```
pin_discount_cleanup -m [close|delete] [-n days] [-d date] [-v] [-t] [-help]
```

Parameters

-m close|delete
Specifies whether to delete discounts when they are canceled:

- **close**

  Changes the status of all active, expired discounts to **canceled** without deleting the discounts.

- **delete**

  Deletes all expired discounts.

-n *days*
The number of days prior to **-d** *date* for which the expired discounts are retained. The utility changes the status to **canceled** for the discounts that expired more than **-n** *days* prior to **-d** *date*.

- d *date*
The end date (in the format *MM/DD/YYYY*) of the period in which discounts that expired are retained.
For example, if **-n** is **5** and **-d** is **07/15/2015**, the status of the discounts that expired before 7/10/2015 are changed to canceled.

> **Note:**
>
> - The expiry date cannot be greater than the current date. For instance, in the example above, if 7/10/2015 is greater than the current date, **pin_discount_cleanup** returns an error. Similarly, if only **-d** is specified, and is greater than the current date, **pin_discount_cleanup** returns an error.
> - If neither -**n** nor **-d** parameter is specified, the current date is used.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter -v at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace filename.log with the name of the log file:
> **pin_discount_cleanup** *other_parameters* **–v** > *filename*.**log**

-t
Displays the number of records processed (the number of discounts that were canceled.

-help
Displays the syntax and parameters for this utility.

Results

To check results of running this utility, look in the log file (normally **default.pinlog**) for error messages. The log file is located in the directory from which the utility was started or in a directory specified in the utility's configuration file (**pin.conf**).

load_pin_snowball_distribution

Use this utility to load snowball discount distribution rules into the **/config/snowball_distribution** object in the BRM database. You define how snowball discounts are distributed in the **pin_snowball_distribution** file in *BRM_home***/sys/data/pricing/example**.

> **Note:**
>
> This utility overwrites existing distribution rules. If you are updating distribution rules, you cannot load new distribution rules only. You must load a complete set of distribution rules each time you run this utility.

Location

*BRM_home***/bin**

Syntax

```
load_pin_snowball_distribution pin_snowball_distribution_file [-d] [-v]
```

Parameters

pin_snowball_distribution_file
The name and location of the file that defines the snowball distribution rules. The default **pin_snowball_distribution** file is in *BRM_home***/sys/data/pricing/example**.
If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

```
load_pin_snowball_distribution BRM_home/sys/data/pricing/example
```

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters. To redirect the output to a log file, use the following syntax:
> **load_pin_snowball_distribution** *other_parameter* **–v >** *filename***.log**

Part IV
Customer and Financial Management

This part describes how to configure customer management and financial functionality in Oracle Communications Billing and Revenue Management (BRM) when using Pipeline Manager. It contains the following chapters:

38

Configuring Balance Monitoring in Pipeline Manager

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Balance Monitoring Manager.

You use balance monitoring to enable your customers to monitor the balance of a group of accounts. Balance monitoring collects the balance impacts for a specified group of accounts and services, and notifies customers when their balance is too high. Customers and CSRs can also access the group's total balance at any time by using a custom client interface.

For information about balance monitoring, see *BRM Accounts Receivable*.

About Balance Monitoring and Pipeline Rating

Pipeline Manager processes monitored balances as follows:

1. Pipeline modules rate the event and apply any discounts.
2. The FCT_Account module determines whether the event owner belongs to any monitor groups and, if it does, enriches the MONITOR_LIST section of the EDR container with information about each monitor group.

   > **Note:**
   >
   > DAT_AccountBatch stores in-memory the list of monitor groups to which an account or service belongs. The module retrieves this information from the BRM database when you first start Pipeline Manager and when you create, modify, or delete a monitor group. For more information, see "About Synchronizing Monitor Data between Cycle-fee and Pipeline Batch Rating".

3. The FCT_BillingRecord module generates a monitor packet for each monitor group.
   - When a balance packet impacts the event balance group, FCT_BillingRecord creates a MONITOR_PACKET for each monitor group in the monitor list.
   - When a balance packet does not impact the event balance group (for example, the discount share or charge share event balance groups), FCT_BillingRecord calls DAT_AccountBatch to retrieve the monitor group list for the discount share or charge share owner's balance group.

   The MONITOR_PACKET contains an account POID of the balance monitor, the balance group ID from the monitor group, balance element ID, amount, sub-balance impact, and the sub-balance block from the balance packet.
4. The pipeline generates an output file that includes any monitor balance impacts.
5. Rated Event (RE) Loader retrieves the pipeline output file and publishes any monitored balance impact data to the monitor queue.

About Synchronizing Monitor Data between Cycle-fee and Pipeline Batch Rating

Pipeline batch rating and cycle-fee rating determine where to apply balance monitor impacts by accessing each member's list of monitor groups. For cycle-fee rating, this information is stored in each member's **/ordered_balgrp** object in the BRM database. For pipeline batch rating, this information is stored in memory by the DAT_AccountBatch module.

When you create, modify, or delete a balance monitor, BRM updates each member's **/ordered_balgrp** object and then uses Account Synchronization to send the updated information to Pipeline Manager, as follows:

1. BRM updates the member's **/ordered_balgrp** object and generates one of the business events listed in Table 38-1:

Table 38-1 Monitor-Related Actions

Action	Event
Create a new monitor group	**/event/group/sharing/monitor/create**
Modify a monitor group by adding or removing members	**/event/group/sharing/monitor/modify**
Delete a monitor group	**/event/group/sharing/monitor/delete**

2. Account Synchronization processes these business events and publishes them to the Account Synchronization queue.
3. Pipeline Manager retrieves the data from the queue and updates its internal memory:
   - The DAT_Listener module retrieves the data from the queue and passes it to DAT_AccountBatch.
   - DAT_AccountBatch refreshes the monitor group list for all accounts.

Enabling Balance Monitoring in Pipeline Manager

Pipeline Manager determines whether balance monitoring is enabled by using the **BalanceMonitoring** entry from the **multi-bal** instance of the /**config/business_params** object.

You must configure Pipeline Manager to use business parameter settings from the BRM database by performing the following:

- Configuring the DAT_PortalConfig module in your registry file. This module must be listed before all other data modules in the registry file.
- Connecting the DAT_AccountBatch module to DAT_PortalConfig by using the **PortalConfigDataModule** registry entry.

> **Note:**
>
> You must enable balance monitoring in Pipeline Manager to have notification events generated during the batch rating process.

39
Setting Up Pipeline-Triggered Billing

This chapter provides a conceptual overview and instructions on setting up Oracle Communications Billing and Revenue Management (BRM) pipeline-triggered billing.

About Pipeline-Triggered Billing

When you use pipeline batch rating, if an account is currently in the process of being billed, incoming call records are suspended (not rated) for that account until its billing is complete. The number of accounts being billed affects the time it takes to complete the billing process. When you need accounts to be billed quickly so that their new usage can be rated, you can set up Pipeline Manager to trigger billing. This will reduce the number of call records that might need suspending. When Pipeline Manager triggers billing for an account, it is billed in a separate billing process.

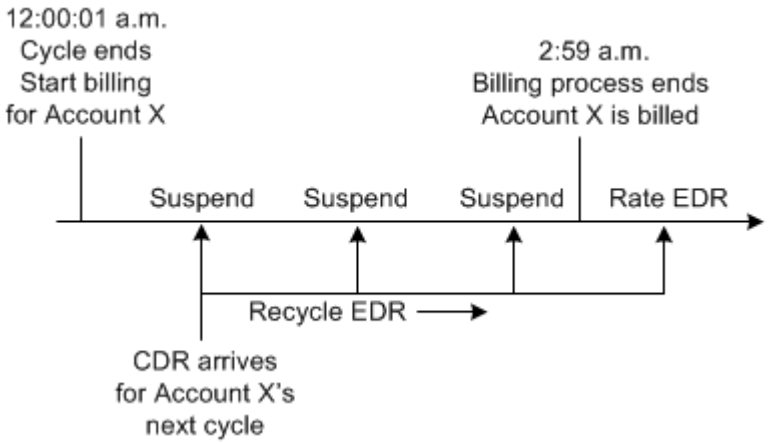
You use pipeline-triggered billing when event data records (EDRs) arrive for the next accounting cycle before the associated accounts have been billed. Pipeline Manager triggers billing for these accounts, enabling the new usage to be rated sooner and reducing the number of EDRs that might need suspending or rerating.

When customers use their services while their accounts are being billed, Pipeline Manager does not rate their usage until the accounts' billing is complete. The BRM billing process can sometimes take several hours. When there is account activity during the billing process, there is a greater possibility that call detail records (CDRs) will need recycling or rerating. When you use pipeline-triggered billing, the accounts are billed in a separate billing process, reducing the billing processing time.

When Pipeline Manager suspends EDRs because it is waiting for the account to be billed, those EDRs must later be recycled for rating. Each time an EDR is recycled, it is suspended until Pipeline Manager receives notification that the billing process for the account is complete.

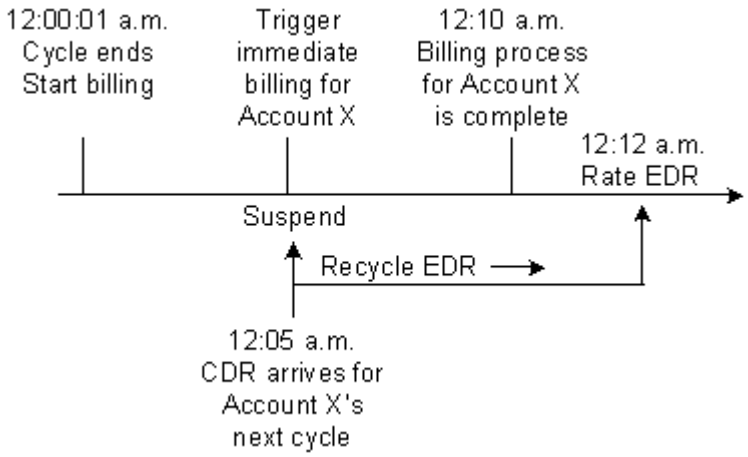
Figure 39-1 shows how a CDR is repeatedly suspended for Account X until the account's billing is complete. In this example, the billing process takes almost three hours.

Figure 39-1 Suspension of CDR Rating During Billing



With pipeline-triggered billing, billing is triggered for Account X when the first new CDR arrives, as shown in Figure 39-2. The account is billed in a separate billing process, reducing the processing time. Pipeline Manager can then rate the recycled CDR and new CDRs that arrive for the next accounting cycle for that account:

Figure 39-2 Pipeline-Triggered Billing



> **Note:**
>
> Performance is affected by the number of accounts that need pipeline-triggered billing. If too many accounts are triggered for billing by Pipeline Manager, there is no performance advantage.

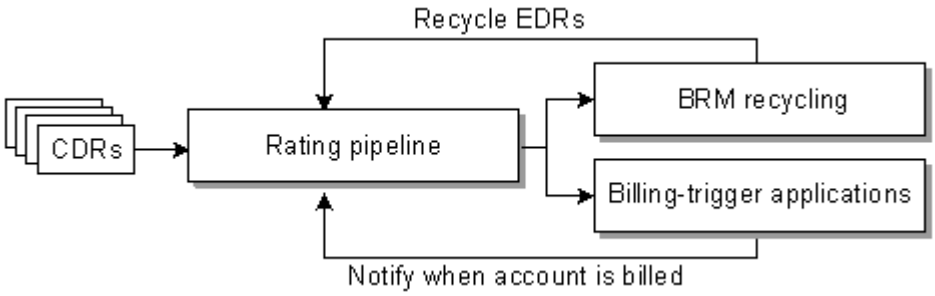
Pipeline-Triggered Billing Components

Pipeline-triggered billing comprises several components that work together to bill accounts:

- **Pipeline Manager modules** flag and route EDRs for accounts that require billing. See "About the Pipeline-Triggered Billing Modules".
- **BRM recycling** suspends EDRs flagged to trigger billing until the accounts are billed. The EDRs are periodically recycled. When the accounts are billed, the recycled EDRs can be rated and are no longer suspended. See "About Suspending Billing-Trigger EDRs".
- **Billing-trigger applications** bill the triggered accounts. These applications include a billing batch handler (BillHandler) and the BRM billing utility. When the accounts are billed, BRM notifies Pipeline Manager that billing for the accounts is complete. See "About BillHandler".

Figure 39-3 shows the relationships among Pipeline Manager, BRM standard recycling, and the billing-trigger applications.

Figure 39-3 Pipeline Manager, Recycling and Billing Applications Relationship



How Pipeline Manager Triggers Billing

The following steps describe the entire process of pipeline-triggered billing:

1. BRM runs billing when the accounting cycle ends.
2. An EDR enters a pipeline for processing.
3. The FCT_TriggerBill module checks if the EDR belongs to the next accounting cycle and if partial billing has not yet been triggered for the account. When both of these conditions are true, it sends the EDR to the billing-trigger output stream.
4. The FCT_Reject module sends the EDR to the suspense output stream.

   For more information, see "About the Pipeline-Triggered Billing Modules".
5. When the EDR reaches the output modules, the data in the EDR takes two routes:

   Suspend EDR:

   a. The suspense output module suspends the EDR by sending it to Suspended Event (SE) Loader. SE Loader stores the EDR in the BRM database.

   b. The **pin_recycle** utility retrieves the EDR from the BRM database and sends it back through the rating pipeline. The EDR begins the cycle again: If billing for the account is complete, the EDR is rated. If billing is not complete, the EDR is again suspended.

      For more information, see "About Suspending Billing-Trigger EDRs".

   Trigger billing:

   a. The pipeline billing-trigger output module creates a file containing the account and bill units associated with the EDR.

   b. The billing-trigger applications retrieve the file and bill the account associated with the EDR.

      For more information, see "About BillHandler".

   c. BRM uses the Account Synchronization Data Manager (DM) to notify Pipeline Manager that the account is billed.

   d. When the recycled EDR is sent back through a rating pipeline (through the Suspend EDR route), the EDR is rated because the account has been billed.

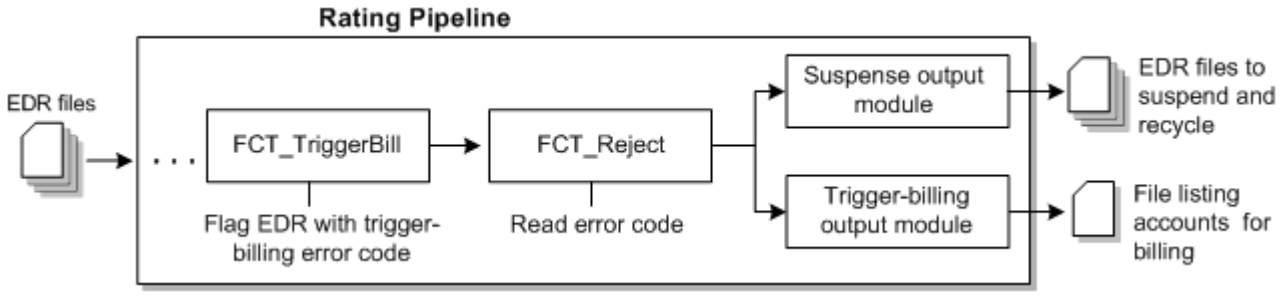
About the Pipeline-Triggered Billing Modules

Pipeline Manager flags EDRs associated with accounts that require immediate billing and sends them to the appropriate output streams. Pipeline-triggered billing uses the following modules:

- The FCT_TriggerBill module, which determines whether EDRs should trigger billing based on the accounting cycle date and billing state. To trigger billing, it sets a billing-trigger error code (**Awaiting billing of account**) in the EDRs. To flag the EDRs for recycling, it sets a billing-trigger recycle key value (**Trigger_Billing**).
- The FCT_Reject module, which detects the billing-trigger error code and sends the EDR to the suspense output stream.
- The billing-trigger output module, which creates a file containing the accounts and bill units for billing and sends the file to a separate directory.
- The suspense output module, which adds the EDRs that trigger billing to an output file. The EDRs are loaded into the BRM database to be suspended and recycled.

Figure 39-4 shows the path that EDRs take when they are flagged to trigger billing:

Figure 39-4 EDR Path for Triggered Billing



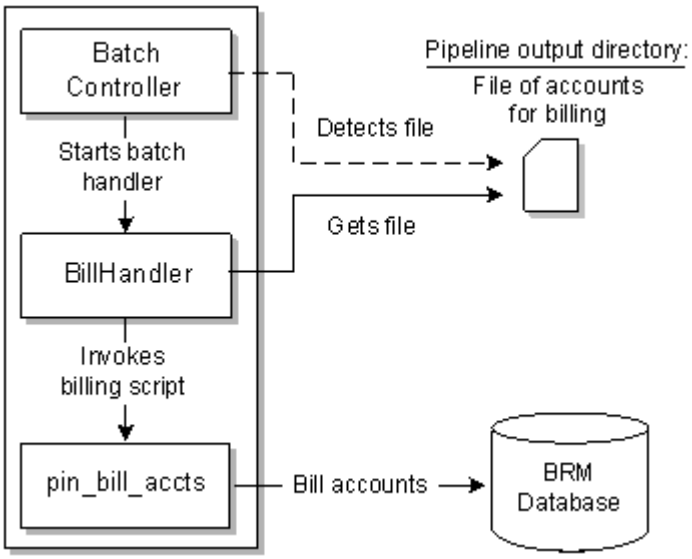
About BillHandler

BillHandler bills the accounts whose EDRs are flagged to trigger billing. BillHandler is used with the following applications:

- **Batch Controller,** which watches for billing-trigger files output by a pipeline. When a file is present, Batch Controller starts BillHandler.
- **The pin_bill_accts billing utility,** which bills the accounts and bill units. BillHandler starts the billing utility.

Figure 39-5 shows the batch handler billing process:

Figure 39-5 Batch Handler Billing Process



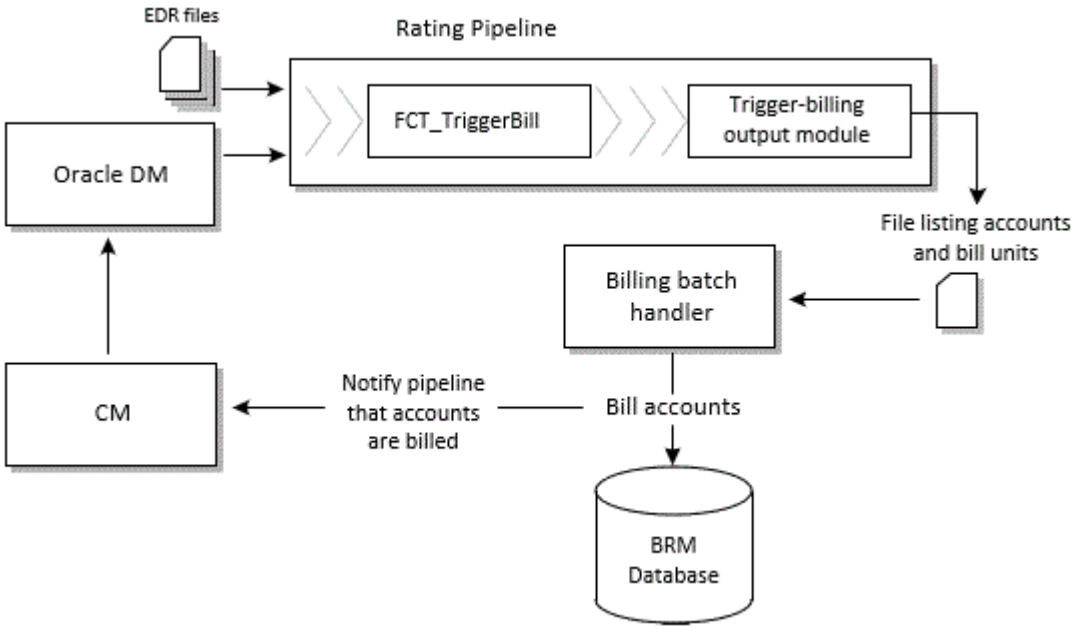
Overview of the Immediate Billing Process

To bill accounts identified for immediate billing, the following actions are performed:

1. Batch Controller starts BillHandler when a file is present in the pipeline billing-trigger output directory.
2. BillHandler reads the accounts and bill units in the file and passes them to the **pin_bill_accts** billing utility to be billed.
3. The **pin_bill_accts** billing utility creates the bills and updates the account information in the BRM database.
4. When the accounts are billed, BRM notifies Pipeline Manager by sending a business event to the Account Synchronization DM.

Figure 39-6 shows an overview of the processes required to bill accounts that are identified for immediate billing:

Figure 39-6 Immediate Accounts Billing Process



About Suspending Billing-Trigger EDRs

EDRs that are flagged to trigger billing are not rated and must be periodically recycled until the accounts are billed. Pipeline-triggered billing uses BRM standard recycling to suspend and recycle these EDRs.

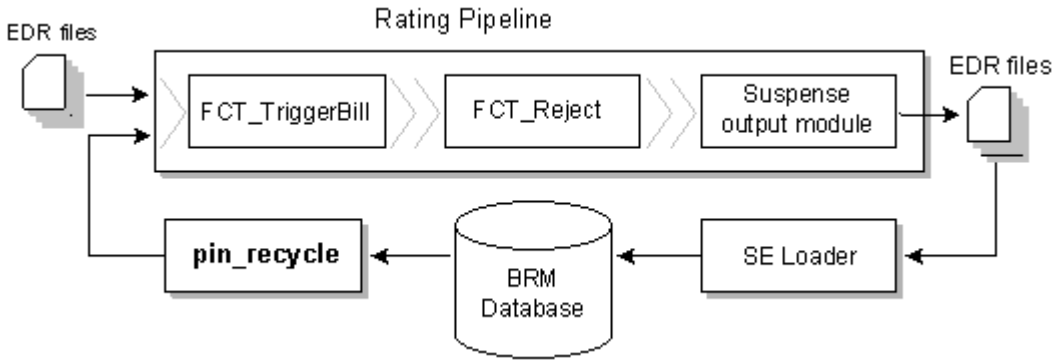
You perform the following actions to suspend and recycle EDRs:

1. When EDRs contain the **Awaiting billing of account** error code, the FCT_Reject module sends them to the suspense output stream.
2. SE Loader retrieves the EDRs from the output stream and stores them in the BRM database.
3. The **pin_recycle** utility retrieves the EDRs from the BRM database and sends them back through the rating pipeline. To continuously recycle any waiting EDRs, schedule the utility to run periodically.

EDRs are suspended each time they are recycled until the accounts are billed. After the accounts are billed, the EDRs are rated and are no longer sent to the suspense output stream.

Figure 39-7 shows an overview of the processes required to suspend and recycle EDRs flagged to triggered billing:

Figure 39-7 Suspension and Recycling of EDRs Flagged to Trigger Billing



Configuring Pipeline-Triggered Billing

To configure pipeline-triggered billing:

- Configure Pipeline Manager. See "Setting Up Pipeline Manager to Trigger Billing".
- Configure Batch Controller. See "Setting Up the Billing Batch Applications".
- Configure Pipeline Manager recycling. You can use either standard recycling or Suspense Manager with pipeline-triggered billing.

Setting Up Pipeline Manager to Trigger Billing

To set up Pipeline Manager to trigger billing:

1. Configure the FCT_TriggerBill module.

   Use the **TriggerBillCreateStream** entry in the module registry to specify the billing-trigger output module.

2. Configure the billing-trigger output stream in the OUT_GenericStream module.

   The default output grammar file is **TriggerBilling_OutGrammar.dsc**.

   The following is an example of the registry entries for the billing-trigger instance of the OUT_GenericStream and EXT_OutFileManager modules:

```
TriggerBillCreateOutput
{
   ModuleName = OUT_GenericStream
   ProcessType = RATING_PIPELINE
   EventType = /event/delayed/session/telco/gsm
   Module
   {
      Grammar = ./formatDesc/Formats/TriggerBill/TriggerBilling_OutGrammar.dsc
      DeleteEmptyStream = True
      OutputStream
      {
         ModuleName = EXT_OutFileManager
         Module
         {
            OutputPath            = ./data/TriggerBill
            OutputPrefix          = trigger_billing
            OutputSuffix          = .tb
            TempPrefix            = .
```

```
            TempDataPath          = ./data/TriggerBill
            TempDataPrefix        = trigger.billing.tmp.
            TempDataSuffix        = .data
            AppendSequenceNumber  = True
            Replace               = True
         }
      }
   }
}
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

3. Add the format and mapping files to the DataDescription registry.
   - The default **StreamFormats** file is **TriggerBilling.dsc**.
   - The default **OutputMapping** file is **TriggerBilling_OutMap.dsc**.

   The following is an example of the billing-trigger entries in the DataDescription registry:

```
DataDescription
{
   Standard
   {
   ModuleName = Standard
   Module
   {
   StreamFormats
   {
      TRIGGERBILL_CREATE_OUTPUT =./formatDesc/Formats/TriggerBill/TriggerBilling.dsc
   }
   . . .
   OutputMapping
   {
      TRIGGERBILL_CREATE_OUTPUT =                          ./formatDesc/Formats/TriggerBill/
TriggerBilling_OutMap.dsc
   }
   }
   }
}
```

Setting Up the Billing Batch Applications

To set up the billing batch applications for pipeline-triggered billing:

- Configure batch controller to start BillHandler. See "Configuring Batch Controller to Start BillHandler".
- Configure BillHandler. See "Configuring BillHandler".

Configuring Batch Controller to Start BillHandler

Use Batch Controller to start BillHandler. You configure Batch Controller to start BillHandler when it detects a file in the pipeline billing-trigger output directory.

To configure Batch Controller:

1. Open the *BRM_home*/**apps/batch_controller/Infranet.properties** file, where *BRM_home* is the directory in which BRM is installed.
2. Add the entries shown in Table 39-1 for BillHandler:

Table 39-1 BillHandler Entries

Entry	Description
batch.random.events	Specify the event identifier. For example: **batch.random.events = Bill_Handler_file** For pipeline-triggered billing, this event is the appearance of a billing-trigger file that is output by a pipeline.
event_name.**name**	Specify a description for the event identifier. For example: **Bill_Handler_file.name = File passed to BillHandler**
event_name.**file.location**	Specify the full path to the pipeline billing-trigger output directory. This is the directory where the billing-trigger output module deposits the file of accounts to be billed.
event_name.**file.pattern**	Specify the billing-trigger output file name. When Batch Controller detects a file with this name, it starts BillHandler. **Tip:** You can use an asterisk (*****) to represent zero or more characters in the file name. No other wildcards are supported. For example: **Bill_Handler_file.pattern = *.out**
event_name.**handlers**	Specify BillHandler identifier. For example: **Bill_Handler_file.handlers = BillHandler**
handler_name.**name**	Specify a description for the BillHandler identifier. For example: **BillHandler.name = Bill Handler that executes pin_bill_accounts**
handler_name.**max.at.lowload.time** *handler_name*.**max.at.highload.time**	Specify the number of BillHandler instances that can run concurrently during periods of low-load and high-load usage. Typical default settings are **6** at low load and **3** at high load.
handler_name.**start.string**	Specify the command that starts BillHandler. The default is *BRM_home*/**apps/pin_bill_handler/BillHandler.pl**.

3. Save and close the file.

Configuring BillHandler

BillHandler retrieves the pipeline-triggered billing output file and sends the account and bill units to the billing utility. After the accounts and bill units are billed, BillHandler deposits the input file to a separate directory.

To configure BillHandler:

1. Open the *BRM_home*/**apps/pin_bill_handler/BillHandler_config.values** file.
2. Edit the entries listed in Table 39-2:

Table 39-2 Configuration Values for BillHandler

Entry	Description
$FILETYPE	Specify the EDR file-name pattern. For example, ***.txt.bc**. **Note:** The asterisk (*****) represents zero or more characters in the file name. No other wildcards are supported. Batch Controller runs BillHandler for each file with a name that matches this pattern.
$HANDLER_DIR	Specify the full path to the directory containing BillHandler, log, input, output, and other files. The default is *BRM_home*/**apps/pin_bill_handler**.
$pinBillActDir	Specify the full path to the directory containing the **pin_bill_accts** billing utility.
$STAGING	Specify the full path to the BillHandler input file location. **Note:** This is typically the same location specified for **$HANDLER_DIR**. You configure this location as the output directory.
$PROCESSING	Specify the full path to the directory from which the billing-trigger files are processed. The default is **$pinBillActDir**.

For information about other entries, see the **BillHandler_config.values** file.

3. Save and close the file.

Running the pin_bill_accts Utility

When Batch Handler runs the **pin_bill_accts** utility, it uses the **-from_file** parameter. This parameter specifies that the accounts to bill will be read from a file, and the associated file name. The input file is passed to the **pin_bill_accts** utility by the BillHandler billing batch handler to trigger billing for the accounts specified in the file.

40

Setting Up Revenue Assurance Manager for Pipeline Batch Rating

This chapter describes how to configure Oracle Communications Billing and Revenue Management (BRM) Revenue Assurance Manager to collect revenue assurance data from pipeline batch rating.

About Revenue Assurance

You use Revenue Assurance Manager to verify the end-to-end completeness, accuracy, and integrity of BRM billing and pipeline batch rating results. You can analyze revenue assurance data to find revenue leakage in your system.

You obtain revenue assurance data by auditing the results of these processes:

- **Billing:** Revenue Assurance Manager provides statistics such as the number of accounts billed or invoiced, the total revenue, and the number of records that were successfully billed or failed to be billed.
- **Pipeline batch rating:** Revenue Assurance Manager provides statistics such as total duration and charges, retail and wholesale amounts, and total discount amounts.

About Collecting Revenue Assurance Data from Pipeline Batch Rating

You can collect revenue assurance data to analyze the effect of pipeline rating on Event Data Records (EDRs). You configure Revenue Assurance Manager to collect statistics on EDRs at various points in your pipelines, and then compare those statistics to see how a batch of EDRs changes as it is processed.

The statistics collected can include:

- The number of EDRs in the batch
- Retail charged amount
- Event wholesale value
- Discounts applied
- Total time usage
- Amount of data transferred
- When a call started
- When a call ended

Revenue assurance data is collected and aggregated at control points that you configure in pipelines. You establish control points by adding the FCT_AggreGate module at appropriate pipeline locations. You determine the data to be collected by specifying aggregation scenarios used by the module.

You can configure related control points into flows that enable you to examine data for a batch sequentially. You can also link pairs of control points to see original and current values.

FCT_AggreGate outputs aggregated data into flat files. You configure the Batch Controller to send these flat files to the Universal Event (UE) Loader as they are created. UE Loader parses the flat files and then calls opcodes to load the information into the database as **/process_audit/batchstat** objects.

Collecting revenue assurance data from pipeline batch rating involves these tasks:

- Configuring event notification to capture data on written-off EDRs and set up revenue assurance alerts.

  See "About Using Event Notification to Generate Revenue Assurance Data" for more information.

- Configuring control points in your pipelines to determine where revenue assurance data is captured.

  See "About Control Points" for more information.

- Associating an aggregation scenario with each of your control points to determine how revenue assurance data is organized. Revenue Assurance Manager includes preconfigured aggregation scenarios that group revenue assurance statistics by using different fields.

  See "About Aggregation Scenarios" for more information.

- Linking pairs of control points related to rating, rerating, and written-off EDRs. Revenue Assurance Center uses these linked pairs to establish the original and current values for a set of EDRs.

  See "About Linking Pairs of Rating, Rerating, and Written-Off Control Points" for more information.

- Defining flows, which are ordered lists of control points that Revenue Assurance Center uses to display revenue assurance data. Flows allow you to track revenue assurance data for EDR batches at various stages during pipeline processing.
- Configuring Universal Event (UE) Loader to load revenue assurance data into the database.
- Configuring alerts to be sent when revenue assurance data passes a threshold that you set.

About Using Event Notification to Generate Revenue Assurance Data

Revenue Assurance Manager uses event notification to collect data on written-off EDRs and set up revenue assurance alerts.

The following events are generated specifically to facilitate the revenue assurance event notification process:

- **/event/notification/suspense/writeoff**: When suspended EDRs are written off, Suspense Manager generates this event. By default, when this event occurs, the PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY opcode is called.
- **/event/notification/ra_threshold**: When specified conditions for producing revenue leakage alerts occur, the **load_pin_config_ra_thresholds** utility generates this event. By default, when this event occurs, the PCM_OP_PROCESS_AUDIT_POL_ALERT policy opcode is called.

About Control Points

You establish control points in batch pipelines to determine where Revenue Assurance Manager collects data. You can configure control points in locations that enable you to compare data from different stages in the rating processes.

You define a control point by adding the FCT_AggreGate module to the pipeline in the desired location and specifying the control point name in the module registry. Each control point must have a unique name that describes its purpose. For example, in a rating pipeline, you could have control points named Rating and CP_After_Rating.

Each control point is associated with an aggregation scenario that specifies the data to be collected and how it should be organized.

You add related control points to *flows*. In Revenue Assurance Center, you can view data from all the control points in a flow. This enables you to follow the progress of EDR batches of EDRs through the pipeline. See "About Flows" for more information.

About Aggregation Scenarios

Each control point requires an aggregation scenario to specify the data that the control point collects and how the data will be organized.

An aggregation scenario specifies:

- The EDR fields to collect data from. For example, a scenario can collect information from the discount amount or duration fields.
- Aggregation functions to apply to the data. For example, you can add data together or average it.
- Fields by which to group the data collected. For example, you can group data by service type. In this case, the data for each batch is grouped by the type of service, such as TEL, SMS, or GPRS.

You can also collect revenue assurance data on more than one grouping field. For example, in the BatchStat_SvcType_Status scenario, the grouping fields are Service Type and EDR Status. The revenue assurance data collected is the EDR status such as Duplicate, Rejected, or Successful for each service type.

You specify the scenario to use when you define a control point in the FCT_AggreGate module registry. Scenarios can be used by any number of control points.

Revenue Assurance Manager provides a number of preconfigured scenarios. These scenarios are suitable for use in a production system. You can also create new scenarios if necessary.

About Linking Pairs of Rating, Rerating, and Written-Off Control Points

Certain types of control points must be linked to establish *original* and *current* values displayed in Revenue Assurance Center. You should link these types of control points:

- A rating control point to a rerating control point.
- A rating control point to a writeoff control point.
- A rerating control point to a writeoff control point.

When pairs of control points are linked, Revenue Assurance Center shows the linked control points in the same area. Values for the first control point are marked as *original* values, and the last control point as *current* values.

About Flows

A flow is an ordered set of related control points that you group together for convenient searching and viewing in Revenue Assurance Center. You can add any number of control points to a flow, from any pipelines that are relevant.

You use Revenue Assurance Center to display data for flows. Each control point appears in its own area. In this example, you can see two control points (P1_END and CP_AfterRating) from Flow12. The second control point shows the effect that rating had on a small batch of EDRs.

About Using UE Loader to Load Revenue Assurance Data

In order to view batch pipeline revenue assurance data, you must first load that data into **/process_audit** objects in the BRM database. You use Universal Event (UE) Loader to load the data.

The UE Loader loads revenue assurance data into the following **/process_audit** objects:

- **/process_audit/pipeline** objects store information about processed EDRs.
- **/process_audit/batchstat** objects store information about the revenue assurance data collected for specific scenarios.

You can configure the BRM Batch Controller to probe automatically for revenue assurance data files and then call UE Loader to load them into the database. You can also decide to load the data periodically by using **cron** or a similar program.

About the Revenue Assurance Data Collected in Rated Event Loader

Revenue Assurance Manager collects data from Rated Event (RE) Loader. BRM uses RE Loader to load events that have been rated or rerated with pipeline batch rating.

The revenue assurance data collected from RE Loader include:

- Total revenue generated by the batch.
- Total number of EDRs loaded.

You can view the RE Loader data in Revenue Assurance Center or in Revenue Assurance reports.

When the rated EDRs are loaded in RE Loader, either all the EDRs are loaded, or they all fail. There is no possibility of some records getting loaded and some records failing. In Revenue Assurance Manager, the data collected for RE Loader is not grouped by status or any other fields.

You must load the **CollectProcessAuditForIREL.sql** file to enable the collection of revenue assurance data from RE Loader.

About Collecting Revenue Assurance Data on Written-Off EDRs

You can use Revenue Assurance Manager to view statistics about the number of EDRs that have been written off.

When an EDR is written off through Suspense Manager, an **/event/notification/suspense/writeoff** event is generated. You configure event notification so that Revenue Assurance Manager collects data every time such an event occurs. The data collected from a written-off EDR includes the original batch ID and the number of EDRs that were written off in that batch of EDRs.

Configuring Revenue Assurance Manager

To set up Revenue Assurance Manager for pipeline batch rating, you need to complete the following tasks:

- Configure event notification to notify Revenue Assurance Manager when events occur.

  See "Configuring Event Notification" for more information.
- Choose scenarios that determine how the revenue assurance data is grouped.

  See "Selecting Aggregation Scenarios" for more information.
- Identify control points for pipeline data collection.

  See "Identifying Control Point Locations for Revenue Assurance Data" for more information.
- Configure FCT_Aggregate to use your aggregation scenarios and control points.

  See "Configuring the FCT_AggreGate Module to Collect Revenue Assurance Data" for more information.
- Configure SimpleSample files to map the batchIDs of the EDRs.

  See "Configuring SimpleSample Files" for more information.
- Add related sets of control points to flows.

  See "Adding Control Points to Flows" for more information.
- Link control points for rating, rerating, and written-off EDRs.

  See "Linking Rating, Rerating, and Write-Off Control Points " for more information.
- Configure Universal Event (UE) Loader to load revenue assurance data into the database.

  See "Setting Up UE Loader to Load Revenue Assurance Data into the Database" for more information.
- Configure Batch Controller to call UE Loader.

  See "Setting Up Batch Controller to Call UE Loader" for more information.

Configuring Event Notification

Revenue Assurance Manager uses event notification to collect data on written-off EDRs and set up revenue assurance alerts.

Before you can use Revenue Assurance Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them.
2. Ensure that the merged file includes the entire event notification list in the *BRM_home***/sys/data/config/pin_notify_ra** file.
3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list.

4. (Optional) If necessary to accommodate your business needs, create custom opcodes for event notification to trigger.
5. Load your final event notification list into the BRM database.

> **Note:**
>
> In addition, your business needs may require that you use event notification to call opcodes when other objects are created in the BRM database.

Selecting Aggregation Scenarios

Aggregation scenarios determine what revenue assurance data is collected from pipelines and how that data is grouped.

You may be able to use one or more of the preconfigured scenarios supplied with Revenue Assurance Manager. Each scenario groups data differently.

If none of the preconfigured aggregation scenarios satisfies your business requirements, you can create your own.

You must associate aggregation scenarios with control points in the registry of the FCT_Aggregate module.

Loading Scenarios into the Pipeline Manager Database

You must load aggregation scenarios into the Pipeline Manager database before you can use them. Scenarios that you create are loaded into the database by Pricing Center at the time of creation.

To load scenarios into an Oracle Pipeline Manager database, run the following command against the Pipeline Manager database from the *pipeline_home***/database/oracle/scripts** directory:

```
% sqlplus user@databaseAlias
Enter password: password

SQL> RevenueAssurance_Scenarios.sql
```

where:

- *pipeline_home* is the directory in which you installed Pipeline Manager.
- *user* is the Pipeline Manager user ID.
- *password* is the Pipeline Manager user password.
- *database* is the Pipeline Manager database alias.

Identifying Control Point Locations for Revenue Assurance Data

You configure control points in pipeline locations where you want to collect revenue assurance data. You associate an aggregation scenario with the control point to determine the data that is collected.

The exact locations where you place control points depend on the data you are collecting. It is often useful to insert control points before and after a critical point in the pipeline. For example,

you can insert a control point before and after rating so that you can analyze the impact on EDRs.

You place a control point in a pipeline by inserting an instance of the FCT_AggreGate module at the desired point in the pipeline. You specify the control point ID and aggregation scenario in the module registry. See "Configuring the FCT_AggreGate Module to Collect Revenue Assurance Data" for more information. Each control point ID must be unique system-wide.

Configuring the FCT_AggreGate Module to Collect Revenue Assurance Data

You insert the FCT_AggreGate module at locations that you identify to collect revenue assurance data. You define a control point and associated aggregation scenario in the registry of each FCT_AggreGate module that you insert.

> **Note:**
>
> If an aggregation scenario requires one or more iScripts, you must specify them in the Function Pool section of the pipeline registry. See "Using iScripts to Derive Grouping Fields" for more information.

You also configure other options in the module registry, including details about control and result files. These are standard options unrelated to revenue assurance.

To configure an FCT_AggreGate module for revenue assurance:

1. Add the FCT_AggreGate module to the pipeline registry at the desired location.
2. Configure the module for revenue assurance data:
   - In the Scenarios section of the registry, create a block for the scenario by entering the scenario name. For example, enter **RA_03** to use the Service Type scenario.
   - For the **ControlPointId** parameter, enter an ID for the control point you are defining. The control point ID must be unique and have maximum of 30 characters. For example, if you use the Service Type scenario, you might define the **CP_PostRatingBatchStat_Svctype** control point.
   - For the **TableName** registry entry, enter the name of the scenario that you defined earlier.

     > **Note:**
     >
     > The value of the **TableName** entry is used as the name of the output files for the scenario. Using the scenario name for this entry makes it easier to associate files with the scenarios from which they were created.

   - Add the registry parameter **IncludeProcessingTimestamps** and set it to **TRUE**.
   - Enter values for the standard FCT_AggreGate parameters, such as **Threshold**, **TempDir**, **DoneDir**, **CtlDir**.
   - Enter a semicolon (;) for the FieldDelimiter.

- Add the parameters **IncludeErrorEDRs** and **IncludeInvalidDetailEDRs** and set them to **TRUE**.

> **Note:**
>
> When configuring scenarios that do not use grouping based on the field **EDRStatus**, do not specify the **IncludeErrorEDRs** and **IncludeInvalidDetailEDRs** parameters.

The following example shows FCT_AggreGate configured for control point **CP_PostRatingBatchStat_Svctype** using the Service Type aggregation scenario (**RA_03**):

```
# Aggregation
------------------------------------------------
# AggreGate
{
 ModuleName = FCT_AggreGate
 Module
 {
  Active                    = TRUE
  ScenarioReaderDataModule = ifw.DataPool.ScenarioReader
  Scenarios
  {
   RA_03
   {
    TableName       = RA_03
    Threshold       = 10000
    TempDir         = ./data/aggregate
    DoneDir         = ./data/aggregate
    CtlDir          = ./data/aggregate
    FieldDelimiter  = ;
    ControlPointId  = CP_PostRatingBatchStat_Svctype
    IncludeErrorEDRs            = TRUE
    IncludeInvalidDetailEDRs    = TRUE
    IncludeProcessingTimestamps = TRUE

  ResultFile
  {
   IncludeFormat  = FALSE
   TempSuffix     = .tmp
   DoneSuffix     = .dat
   WriteEmptyFile = FALSE
  }
  ControlFile
  {
   IncludeFormat  = FALSE
   Suffix         = .ctl
   DataFilePath   = TRUE
  }
 }
}
```

Using iScripts to Derive Grouping Fields

Review the description of the scenario you are using to see if it requires any iScripts. Scenarios use iScripts to derive grouping fields such as **EDR Status**, **Revenue Stream**, and **Output Stream**.

If required, you need to specify the iScripts in the FunctionPool section of the registry file, before the FCT_AggreGate module.

- ISC_SetEDRStatus
- ISC_SetOutputStream
- ISC_SetRevenueStream
- ISC_SetRevenueFigures

The following iScript must be placed at the beginning of the pipeline to ensure that the batch ID is inserted before any further processing of the mediation batches.

- ISC_SetAndValidateBatchInfo

Configuring SimpleSample Files

Configure the SimpleSample files to map the batchIDs of the EDRs. This is a mandatory step for Revenue Assurance Manager to work correctly.

You can find the SimpleSample files at: *pipeline_home***/ifw/formatDesc/Formats/Sample**.

To configure the SimpleSample files:

1. Open the **SIMPLESAMPLE_v1.dsc** file using a text editor such as Notepad.
2. Find the following line:

```
SERVICE                    AscString();
```

3. Add the following lines before the aforementioned line:

```
BATCH_ID                   AscString();
ORIGINAL_BATCH_ID          AscString();
SUSPENDED_FROM_BATCH_ID    AscString();
EVENT_ID                   AscString();
```

4. Save and close the file.
5. Open the **SIMPLESAMPLE_v1_InMap.dsc** file using a text editor such as Notepad.
6. Find the following line:

```
"020"                    -> DETAIL.RECORD_TYPE;
```

7. Add the following lines before the aforementioned line:

```
BATCH_ID                -> DETAIL.BATCH_ID;
ORIGINAL_BATCH_ID       -> DETAIL.ORIGINAL_BATCH_ID;
SUSPENDED_FROM_BATCH_ID  -> DETAIL.ASS_SUSPENSEEXT.SUSPENDED_FROM_BATCH_ID;
EVENT_ID                 -> DETAIL.EVENT_ID;
```

8. Save and close the file.
9. Open the **SIMPLESAMPLE_v1_InGrammar.dsc** file using a text editor such as Notepad.
10. Find the following line:

```
edrInputMap( "SIMPLESAMPLE_V1.DETAIL.STD_MAPPING" );
```

11. Add the following lines before the aforementioned line:

```
edrAddDatablock( DETAIL.ASS_SUSPENSE_EXT );
```

12. Save and close the file.

Adding Control Points to Flows

Flows are ordered sets of related control points. You group control points into flows so that you can search for and view them conveniently in Revenue Assurance Center. When you view a flow in Revenue Assurance Center, its control points are displayed in the order in which they were defined.

You define flows in the **pin_config_ra_flows** file and load the flows into the database by using the **load_pin_config_ra_flows** utility. Flows are stored as **/config/ra_flows** objects.

The **load_pin_config_ra_flows** utility overwrites existing flows. You must load a complete set of flows each time you run the utility.

To add control points into flows and load them into the database:

1. Open the *BRM_home*/**sys/data/config/pin_config_ra_flows** file in a text editor.

   This file includes instructions about the syntax to use to add control points to flows.

2. Save and close the file.
3. Use the following command to load your control points into the database:

   ```
   % load_pin_config_ra_flows pin_config_ra_flows
   ```

   If you do not run the utility from the directory in which the file is located, include the complete path to the file; for example:

   ```
   % load_pin_config_ra_flows BRM_home/sys/data/config/pin_config_ra_flows
   ```

> **Tip:**
>
> If you copy the **pin_config_ra_flows** file to the directory from which you run the **load_pin_config_ra_flows** utility, you do not have to specify the path or file name. The file must be named **load_pin_config_ra_flows**.

To verify that the flows were loaded, you can display the **/config/ra_flows** object by using the Object Browser, or use the **robj** command with the **testnap** utility.

Linking Rating, Rerating, and Write-Off Control Points

For Revenue Assurance Center to recognize original and current values for certain types of control points, you must link them and add the links to the BRM database. The control points that require linking include:

- Rating to rerating.
- Rerating to written-off.
- Rating to written-off.

To link control points, you define the links in the **pin_config_controlpoint_link** file and then load them into the BRM database by using the **load_pin_config_controlpoint_link** utility. Links are stored in the **/config/link_controlpoint** object in the BRM database. The PCM_OP_PROCESS_AUDIT_CREATE_AND_LINK opcode links **/process_audit/batchstat** objects based on the data in the **/config/link_controlpoint** object.

> **Caution:**
>
> The **load_pin_config_controlpoint_link** utility overwrites existing links. You must load a complete set of links each time you run the utility.

> **Note:**
>
> The **load_pin_config_controlpoint_link** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

To configure your control points into flows and load them into the database:

1. Open the *BRM_home*/**sys/data/config/pin_config_controlpoint_link** file in a text editor. This file includes instructions on how to add control points into flows.
2. Save and close the file.
3. Use the following command to load your control points into the database:

   ```
   % load_pin_config_controlpoint_link pin_config_controlpoint_link
   ```

   If you do not run the utility from the directory in which the file is located, include the complete path to the file; for example:

   ```
   % load_pin_config_controlpoint_link BRM_home/sys/data/config/
   pin_config_controlpoint_link
   ```

   > **Tip:**
   >
   > If you copy the **pin_config_controlpoint_link** file to the directory from which you run the **load_pin_config_controlpoint_link** utility, you do not have to specify the path or file name. The file must be named **pin_config_controlpoint_link**.

To verify that the control point links were loaded, you can display the **/config/controlpoint_link** object by using Object Browser, or use the **robj** command with the **testnap** utility.

Setting Up UE Loader to Load Revenue Assurance Data into the Database

UE Loader takes the revenue assurance data from the FCT_AggreGate output files and loads it into the BRM database. The following sections explain how to configure UE Loader.

Setting Up UE Loader Templates

UE Loader requires that each aggregation scenario have a separate UE Loader event input template in XML format to map the data generated by FCT_AggreGate to the input flist for the opcodes that data. Revenue Assurance Manager provides XML templates for each of the preconfigured scenarios.

Setting Up Batch Controller to Call UE Loader

Batch Controller runs programs such as UE Loader either when a specific event occurs or automatically at timed intervals.

Most implementations use the occurrence-driven execution feature of Batch Controller to run UE Loader whenever FCT_AggreGate creates output files with revenue assurance data.

> **Note:**
>
> If you set up Batch Controller to load scenario data files periodically rather than by occurrence, make sure the files are loaded frequently. Revenue Assurance Center uses the time when records were loaded into the database when it searches for data in time range. For the revenue assurance data to be meaningful, it should be loaded into the database soon after creation.

Setting Up Revenue Assurance Manager to Collect Data on Written-Off EDRs

For written-off EDRs, Revenue Assurance Manager collects the original batch ID and the number of EDRs that were written off in the batch.

To collect revenue assurance data on written-off EDRs, do the following:

- Enable event notification for Revenue Assurance Manager.
- Set a parameter in the Connection Manager **pin.conf** file to configure the control point ID to collect the revenue assurance data on written-off EDRs. You can change this control point ID.

About Aggregation Scenarios

Aggregation scenarios specify the data that is collected by Revenue Assurance Manager from pipeline batch rating. You must specify an aggregation scenario at every control point you set up in the pipeline.

Revenue Assurance Manager includes scenarios that are suitable for use in a production system. These scenarios group revenue assurance data in different ways. See "Data Fields Collected by All Scenarios" for a list of the fields from which the scenarios collect data. See "Fields Used to Group Scenario Data" for a list of the fields by which the scenarios can group data.

For example, the EDR statistics for scenario **RA_03** are grouped by service type, such as TEL, SMS, or GPRS. Data, such as event count, retail charged amount, and so on, is aggregated for each service type.

Most of the scenarios group by combinations of grouping fields. For example, scenario **RA_04** groups fields by both service type (TEL, SMS, and so on) and EDR status (duplicate, rejected, written-off or successful). Data is aggregated separately for each combination; duplicate EDRs for the SMS service, for example.

If the preconfigured scenarios do not meet your business needs, you can also create your own.

> **Note:**
>
> Custom scenarios require custom UEL templates and cannot be used with Revenue Assurance Center.

Before you can use the preconfigured scenarios, you must load them into the pipeline database.

Data Fields Collected by All Scenarios

All of the preconfigured aggregation scenarios collect statistics from the EDR fields listed in Table 40-1:

Table 40-1 All Scenarios Data Fields

Field	Description
Event count	Number of EDRs.
Retail charged amount	Retail charged amount collected.
Event wholesale value	Total wholesale amount charged for the EDR, if appropriate.
Discount amount	Discount amount applied.
Duration	Total usage time, in seconds, if appropriate (for example, for a voice call).
Volume sent	Data transferred in bytes, if appropriate (for example, for a GPRS event).
Volume received	Data received in bytes, if appropriate (for example, for a GPRS event).
Earliest call made	The earliest call start timestamp.
Latest call made	The latest call start timestamp.

Fields Used to Group Scenario Data

The EDR data fields in Table 40-2 are available to use for grouping data.

Table 40-2 Grouping Data Fields

Grouping Field	Description
Batch ID	Batch ID of the batch being processed.
Original batch ID	Mediation batch ID of the rerating or recycling EDR batches.
Suspended from batch ID	The batch ID from which the EDR was suspended.
Service type	Type of service, such as TEL, SMS, or GPRS.
Revenue stream	Retail, Wholesale, or Roaming.
Output stream	The output stream of the EDR. This is based on the service type; for example, if the service type is SMS, the output stream is SMS.
EDR status	Success, Suspense, Duplicate, Discard, Rejected, or Skipped.
Suspense code	Suspense reason code assigned to the EDR.

Preconfigured Aggregation Scenario Details

Table 40-3 provides information about the Revenue Assurance Manager preconfigured aggregation scenarios.

Table 40-3 RA Manager Preconfigured Aggregation Scenarios

Scenario/ File Name	Collects Data for an EDR Batch Based On	Grouping Fields (Listed in the Grouping Order)	Storable Class	Installation Point	iScripts Required
RA_01, Batchstatat_simple	An EDR batch.	Batch ID, Original batch ID, Suspended from batch ID	**/process_audit/ batchstat/simple**	Anywhere in a pipeline	None
RA_02, BatchStat_status	EDR Status.	Batch ID, Original batch ID, Suspended from batch ID, EDR status	**/process_audit/ batchstat/status**	Anywhere in a pipeline	ISC_SetEDR Status
RA_03, BatchStat_Svc Type	Service type.	Batch ID, Original batch ID, Suspended from batch ID, Service type	**/process_audit/ batchstat/svctype**	After FCT_Service CodeMap	None
RA_04, BatchStat_Svc TypeStatus	Service type and EDR status. Using this scenario, you can find the number of records that are duplicate, rejected, or successful for each service type.	Batch ID, Original batch ID, Suspended from batch ID, Service type, EDR status	**/process_audit/ batchstat/ svctype_status**	After FCT_Service codeMap	ISC_SetEDR Status
RA_05, BatchStat_RevenueStream	Revenue stream. The revenue streams are: Retail, Wholesale, and Roaming.	Batch ID, Original batch ID, Suspended from batch ID, Revenue stream	**/process_audit/ batchstat/ revstream**	After Post Rating	ISC_SetRevenueStream
RA_06, BatchStat_Revenue Stream_Status	Revenue stream and EDR status.	Batch ID, Original batch ID, Suspended from batch ID, Revenue stream, EDR status	**/process_audit/ batchstat/ revstream_status**	After Post Rating	ISC_SetRevenueStream, ISC_SetEDR Status
RA_07, BatchStat_RevenueStream_SvcType	Revenue stream and service type.	Batch ID, Original batch ID, Suspended from batch ID, Revenue stream, Service type	**/process_audit/ batchstat/ revstream_svctype**	After Post Rating	ISC_SetRevenueStream
RA_08, BatchStat_RevenueStream_ServiceType_Status	Revenue stream, service type, and EDR status.	Batch ID, Original batch ID, Suspended from batch ID, Revenue stream, Service type, EDR status	**/process_audit/ batchstat/ revstream_svctype_status**	After Post Rating	ISC_SetRevenueStream, ISC_SetEDR Status
RA_09, BatchStat_Svc Type_Revenue Stream	Service type and the revenue stream.	Batch ID, Original batch ID, Suspended from batch ID, Revenue stream, Service type	**/process_audit/ batchstat/ revstream_status_svctype**	After Post Rating	ISC_SetRevenueStream
RA_10, BatchStat_ServiceType_RevenueStream_Status	Service type, revenue stream, and EDR status.	Batch ID, Original batch ID, Suspended from batch ID, Revenue stream, Service type, EDR status	**/process_audit/ batchstat/ svctype_revstream_status**	After Post Rating	ISC_SetRevenueStream, ISC_SetEDR Status

Table 40-3 (Cont.) RA Manager Preconfigured Aggregation Scenarios

Scenario/ File Name	Collects Data for an EDR Batch Based On	Grouping Fields (Listed in the Grouping Order)	Storable Class	Installation Point	iScripts Required
RA_11, BatchStat_Outputstream	Output stream.	Batch ID, Original batch ID, Suspended from batch ID, Output stream	**/process_audit/batchstat/outputstream**	Just before output	ISC_SetOutputStream
RA_12, BatchStat_ServiceType_RevenueStream_Outputstream	Service type, revenue stream, and output stream.	Batch ID, Original batch ID, Suspended from batch ID, Output stream, Service type, Revenue stream	**/process_audit/batchstat/svctype_revstream_outputstream**	Just before output	ISC_SetRevenueStream and ISC_SetOutputStream
RA_13, BatchStat_Suspense	Suspense reason code.	Batch ID, Original batch ID, Suspended from batch ID, Suspense code	**/process_audit/batchstat/suspense**	After FCT_Suspense	None
RA_14, BatchStat_ServiceType_RevenueStream_Status_Outputstream	Service type, revenue stream, EDR status, and output stream.	Batch ID, Original batch ID, Suspended from batch ID, Service type, Revenue stream, EDR status, Output stream	**/process_audit/batchstat/svctype_revstream_status_outputstream**	Just before output	ISC_SetRevenueStream, ISC_SetOutputStream, and ISC_SetEDRStatus

Creating New Aggregation Scenarios for Revenue Assurance

Revenue Assurance Manager uses aggregation scenarios to group the data that it collects from the pipeline. If the aggregation scenarios provided with Revenue Assurance Manager do not meet you needs, you can create your own.

> **Note:**
>
> You cannot view data from custom scenarios in Revenue Assurance Center.

To create your own aggregation scenarios:

1. Create an aggregation scenario by using Pricing Center or PCC.
2. Create the appropriate **/process_audit/batchstat** subclasses.

   > **Note:**
   >
   > The new subclasses created must contain an array field of type PIN_FLD_GROUP_DETAILS. The PIN_FLD_ORIGINAL_BATCHID field must also be created in the array.

3. Create the XML templates used by Universal Event (UE) Loader to load the aggregation data generated by the new scenario.
4. Modify the PCM_OP_PROCESS_AUDIT_POL_CREATE_AND_LINK policy opcode to check for duplication of records in the newly created **/process_audit** object.

Tracking EDRs by Using Batch IDs

In pipeline batch rating, each batch file received from the mediation system is assigned a batch ID, which is stored in every EDR derived from the file. Each EDR also receives a unique event ID.

> **Note:**
>
> The **KeepExistingBatchIds** registry entry in FCT_PreSuspense module controls the way batch IDs are set. For details, see "Setting the Default Batch ID Behavior".

During rerating and recycling, the EDR receives a new batch ID, but the original batch ID is retained in a different field. Retaining the original batch ID in the EDR makes it possible to determine the revenue impact of EDRs for each batch that is received from mediation, even if some EDRs are rerated or recycled.

Revenue Assurance Manager uses the following fields to track EDRs as they are processed by pipelines and as they are rerated or recycled:

- DETAIL.BATCH_ID
- DETAIL.ORIGINAL_BATCH_ID
- DETAIL.ASS_SUSPENSE_EXT.SUSPENDED_FROM_BATCH_ID

For example, BRM receives a batch file with batch ID **Mediation087**. All EDRs for events in the file are assigned this batch ID. The batch rating pipeline processes EDRs from this batch loads their data into the BRM database.

Later, some of the EDRs from this batch and a second batch, **Mediation099**, are rerated. During rerating, the two sets of EDRs from different batches are given the new batch ID **ReratingBatch007**. When the individual EDRs are given the new batch ID, their original batch IDs are moved to the ORIGINAL_BATCH_ID field.

Table 40-4 contains selected data from an EDR in the batch after rating:

Table 40-4 Rating EDR Data

Event ID	Duration	Charge	Batch ID	Original Batch ID
189	180	3	Mediation087	Mediation087

Table 40-5 contains the data for the EDR after rerating:

Table 40-5 Rerating EDR Data

Event ID	Duration	Charge	Batch ID	Original Batch ID
189	180	5	ReratingBatch007	Mediation087

Keeping Track of Rejected EDRs by Using Batch IDs

When a batch file is processed, some of its EDRs may not be able to be rated because of missing data or another reason. These rejected EDRs can be processed by Suspense

Manager and recycled back into the rating pipeline. EDRs that are being rerated can also be rejected and sent to recycling.

When Suspense Manager recycles the rerated EDRs back through the pipeline, they receive new batch IDs based on the recycling batch. Their original batch IDs remain to reflect the mediation batch they started in. The suspended-from batch ID field (DETAIL.ASS_SUSPENSE_EXT.SUSPENDED_FROM_BATCH_ID) stores the ID of the batch in which the EDR was rejected. This could be the original batch or a rerating batch.

For example, two batches (MED1 and MED2) are received from the mediation system and processed by the batch rating pipeline. Some EDRs from each of the two batches are rejected and then recycled as part of batch RCL1. In addition, some EDRs from the original two batches are rerated as part of batch RRT1. Some of the EDRs in that rerating batch are rejected and then recycled as part of batch RCL2.

Table 40-6 summarizes how the three different batch ID fields change as EDRs are rated, rerated, and recycled.

Table 40-6 Batch ID Changed Fields

Pipeline Process	Value for BATCH_ID	Value for ORIGINAL_BATCH_ID	Value for SUSPENDED_FROM_BATCH_ID
Rating Batch MED1	MED1	MED1	MED1*[1]
Rating Batch MED2	MED2	MED2	MED2*[1]
Recycle Batch RCL1 (containing suspended EDRs from MED1 and MED2)	RCL1	MED1/MED2	MED1/MED2
Rerating Batch RRT1 (containing EDRs from MED1 and MED2)	RRT1	MED1/MED2	RRT1*[1]
Recycle Batch RCL2 (containing suspended EDRs from RRT1)	RCL2	MED1/MED2	RRT1

1 The value of the suspended-from batch ID is ignored in rating and rerating. Because it is left blank, it is assigned the value of batch ID.

By linking the control point in the original mediation pipeline to the control point in the recycle pipeline that processed the rerated EDRs, you can determine the revenue impact for each of the mediation batches and identify the revenue leakage in your system.

Setting the Default Batch ID Behavior

The **KeepExistingBatchIds** registry entry in FCT_PreSuspense module controls whether the values for batch IDs in EDR records are preserved as originally set by the input module.

- A value of **True** preserves the batch ID in each detail record of the batch input file.
- A value of **False** (the default) sets the batch ID of each record to the batch ID contained in the header record of the batch input file.

41

Setting Up Pipeline Manager Taxation

This chapter describes how to configure your Oracle Communications Billing and Revenue Management (BRM) system to tax events rated by a batch pipeline.

About Pipeline Taxation

BRM can apply taxes to events rated by Pipeline Manager. You can configure your system to tax pipeline events in one of the following ways:

- **During the pipeline rating process**. Taxing events during pipeline rating allows you to generate pipeline output files that include both usage and tax charges. This is useful when output files are not sent through the billing process. For example, when you rate outcollect roaming calls, you rate calls from another carrier's customer and then send the pipeline output files directly to the other carrier for rating and verification.

  > **Note:**
  >
  > You can apply only flat taxes during pipeline rating. For complex taxation, you must defer taxation to the billing process.

- **During the billing process**. Deferring taxation to the BRM billing process allows you to apply complex tax rules using third-party tax software or customized BRM policy opcodes. It also reduces rounding errors because BRM simultaneously taxes all events of the same type. This configuration works best for most wireless applications because it allows complex taxation.
- **During both pipeline rating and billing**. Taxing events during both pipeline rating and BRM billing allows you to:
  - Obtain itemized tax charges for each usage event.
  - Tax non-usage events, such as one-time purchases and cycle-time events that are rated in real-time rather than by Pipeline Manager.

Setting Up Pipelines to Tax Events

You can configure a pipeline to apply flat taxes to events. To use flat taxes during pipeline batch rating, you configure ISC_TaxCalc in your pipelines.

> **Note:**
>
> A pipeline can apply only flat taxes. To use more complex taxation, you must defer taxation to the billing cycle.

ISC_TaxCalc must run *after* the FCT_MainRating module and *before* the FCT_BillingRecord module. If you want to round the tax balance impact, you can optionally run the FCT_Rounding module immediately after ISC_TaxCalc.

Pipeline Manager applies taxes as follows:

1. The pipeline input and function modules process call records.
2. The FCT_MainRating module rates the call record and adds charge packets to the associated charge breakdown record.
3. The ISC_TaxCalc iScript module applies a flat tax.

   See "About Applying a Flat Tax by Using the ISC_TaxCalc iScript " for more information.
4. The FCT_BillingRecord module consolidates the billing charges and flags whether an event was taxed in a pipeline.

   See "About Consolidating Tax Data by Using the FCT_BillingRecord Module" for more information.
5. (Optional) The FCT_Rounding module rounds tax balance impacts.
6. The remaining function and output modules finish processing the call records and generating output files.

See "Configuring Pipelines to Apply Flat Taxes" for information about configuring the pipeline for taxation.

Configuring Pipelines to Apply Flat Taxes

To configure your pipelines to apply flat taxes:

1. Use ISC_TaxCalc in your pipelines.

   See "About Applying a Flat Tax by Using the ISC_TaxCalc iScript " and "About Consolidating Tax Data by Using the FCT_BillingRecord Module" for more information.
2. Configure ISC_TaxCalc.

About Applying a Flat Tax by Using the ISC_TaxCalc iScript

You use the ISC_TaxCalc iScript to apply flat taxes to charges flagged for taxation.

When ISC_TaxCalc processes an event data record (EDR), it determines whether the EDR should be taxed by checking whether the **Tax Treatment** flag was applied to the pipeline charge that applies to the event. If the flag isn't set, ISC_TaxCalc ignores the EDR and passes it on to the next module in the pipeline. If the Tax Treatment flag is set, ISC_TaxCalc:

- Determines which tax rate to use by checking the **taxcodes_map** data stored in the cache, if the **taxcodes_map** information is provided:
  - If the event meets one of the tax criteria, ISC_TaxCalc uses the appropriate tax rate defined in the **taxcodes_map** file.
  - If the event does not meet any tax criteria, ISC_TaxCalc uses the default tax rate specified in the registry file.
- If the **TaxCodeMapFilePath** parameter is not defined, the module retrieves tax code map data from the **/config/taxcodes_map** object.
- Calculates the flat tax for the pre-tax amount.

> **Note:**
>
> The pre-tax amount is the charged amount minus any discounts.

- Populates the tax packet data block in the EDR with the pre-tax amount, tax percentage, and tax balance impact.

About Consolidating Tax Data by Using the FCT_BillingRecord Module

You use the FCT_BillingRecord module to consolidate charge packets, discount packets, and tax packets into an associated BRM billing record in the EDR.

When an EDR is flagged for taxation, FCT_BillingRecord:

- Creates a balance impact packet that consolidates the charge packet and any tax packet data.
- Flags whether ISC_TaxCalc applied taxes to the EDR by using the PIN_DEFERRED_AMOUNT field. It sets this field to **0** when an EDR contains a tax packet and to PIN_AMOUNT when an EDR does not contain a tax packet.

Sample Flat Tax Configurations

This section shows how to set up flat taxes for two sample applications:

- Applying Flat Taxes to Outcollect Roaming Calls
- Applying Flat Taxes to Third-Party Content Usage

Applying Flat Taxes to Outcollect Roaming Calls

Roaming allows customers of one network operator to use their mobile phones in foreign networks. When customers travel to other regions, they can use the services of any network operator with a roaming agreement with their home network operator.

When customers from another network use your network (outcollect roaming), you rate those calls and bill the customer's network operator for this usage. Applying a flat tax during the rating process allows you to pass on local taxes to the other network operator.

To apply a local flat tax to outcollect roaming calls:

1. In PDC, ensure that the tax calculation is enabled in charge offers.
2. Set up a custom flat tax.
3. Configure the ISC_TaxCalc iScript in your outcollect processing pipeline.

   See "Configuring Pipelines to Apply Flat Taxes" for more information.

After the pipeline generates output files that include both usage and tax charges:

- Send one output file to the other network operator for verification and rating.
- Load settlement data into the BRM database and generate invoices for billing the other network operator.

Applying Flat Taxes to Third-Party Content Usage

Wireless carriers allow customers to access third-party content, such as news and sports scores, through mobile phones. Customers are often charged monthly subscription fees to access the content and usage fees when downloading files. Carriers bill their customers for this usage and then remit some of the fees to the third-party provider.

In this scenario, taxes are applied to:

- Usage events during the pipeline batch rating process.
- Cycle and one-time purchase events during the billing process.

To configure your system to apply flat taxes to third-party content usage:

1. In PDC, ensure that tax calculation is enabled in charge offers.
2. Set up your custom tax rates.
3. Configure the ISC_TaxCalc iScript in your pipelines.
4. Use Rated Event (RE) Loader to load your pipeline output files into the BRM database.
5. Configure BRM to perform deferred taxation.

When BRM completes the billing process, you can run the remittance utility and generate a report summarizing the amount owed to each third-party provider. See "Remitting Funds to Third Parties" in *BRM Configuring and Running Billing*.

42

Credit Limit and Threshold Checking during Batch Rating

This document describes how to manage credit limits when using Oracle Communications Billing and Revenue Management Pipeline Manager.

About Credit Limit and Threshold Checking during Batch Rating

During batch rating, Pipeline Manager determines whether a customer's *noncurrency* balance has breached a credit threshold. Pipeline Manager modules compare a customer's balance against noncurrency credit threshold values and then compile a list of events that crossed thresholds. A utility loads the list from Pipeline Manager into BRM.

> **Note:**
>
> Pipeline rating calculates balance impacts and credit threshold breaches immediately. However, the utility that loads the balance impacts into the BRM database is different from the utility that loads credit threshold breaches into BRM. This means that a notification for a credit threshold breach might be sent before the customer's balance is updated in the BRM database. Therefore, the customer balance shown on the threshold breach notification might temporarily differ from the balance in the customer's account.

The following pipeline modules are responsible for checking credit thresholds during the batch rating process:

- DAT_PortalConfig retrieves credit profile information from the BRM database and stores it in its internal memory.
- DAT_BalanceBatch retrieves the customer's current credit profile from the DAT_PortalConfig module, compares the customer's updated balance to the credit profile, and notifies FCT_ApplyBalance if a credit threshold or credit limit has been breached.
- FCT_ApplyBalance updates the customer's current balance in the DAT_BalanceBatch internal memory and, if a credit threshold has been breached, writes the event detail record (EDR) into an internal list.

  To help you identify the exact event that caused the threshold breach, FCT_ApplyBalance adds the following event attributes to each threshold breach notification:

  - Event type (DETAIL.EVENTTYPE)
  - Call originator (DETAIL.A_NUMBER)
  - Call destination (DETAIL.B_NUMBER)

  At the end of the transaction, FCT_ApplyBalance writes all of the EDRs from the internal list to an XML output file. See "About the Format of the XML Output File Name".

After the XML output file is generated, Batch Controller uses the **load_notification_event** utility to load the notification events into BRM. See "About Loading Notifications from Pipeline Manager to BRM ".

About the Format of the XML Output File Name

Pipeline Manager generates output XML files using the following file name format:

OutputPrefix_PipelineName_StreamName_TransactionID_SequenceNumber

where:

- *OutputPrefix* is the prefix name specified in the **OutputPrefix** registry entry.
- *PipelineName* is the name of the individual pipeline, such as ALL_RATE.
- *StreamName* is based on the **unitsPerTransaction** parameter maintained at the input level of the pipeline. A value of **1** means that one input file generates one output file. A value greater than **1** means that multiple input files are merged into one output file. *StreamName* is replaced with the name of the last file merged into the output file. For example, if **unitsPerTransaction** was **3** and the files merged into the output file were **file01**, **file02**, and **file03**, *StreamName* would be replaced with **file03**.
- *TransactionID* is the system-generated transaction ID number.
- *SequenceNumber* signifies the order in which FCT_ApplyBalance created the XML output file within a transaction. For example, the first XML output file has a sequence number of 1, the second output file has a sequence number of 2, and so on.

For example:

```
balancenotification_ALL_RATE_file01_776_1
```

About Loading Notifications from Pipeline Manager to BRM

Notifications from Pipeline Manager are loaded into the BRM database by the **load_notification_event** utility. This utility is launched automatically by Batch Controller whenever Pipeline Manager writes an output XML file to a specified directory.

When Pipeline Manager writes an output XML file to the specified directory, notifications are loaded as follows:

1. Batch Controller launches the SampleLoadHandler batch handler.
2. SampleLoadHandler moves the output XML file to a processing directory and runs the **load_notification_event** utility.
3. **load_notification_event** converts the notification events in the output XML file into flist format and loads them into BRM.
4. The BRM event notification system sends an alert to the customer.
5. SampleLoadHandler determines whether the utility successfully loaded the notification events into BRM:
   - If the utility was successful, SampleLoadHandler moves the output XML file from the processing directory to the archive directory.
   - If the utility failed, SampleLoadHandler moves the output XML file from the processing directory to the reject directory.

You must configure Batch Controller, the batch handler, and the **load_notification_event** utility to process the output XML files. See "Configuring Batch Controller to Run load_notification_event".

Setting Up the Batch Rating Process to Perform Threshold Checking

To set up your system to perform threshold checking during the batch rating process:

- Enable threshold checking. See "Enabling Threshold Checking in Pipeline Manager".
- Configure Batch Controller to run the **load_notification_event** utility. See "Configuring Batch Controller to Run load_notification_event".
- Configure Pipeline Manager to check thresholds. See "Configuring Pipeline Manager to Perform Threshold Checking".

Enabling Threshold Checking in Pipeline Manager

By default, threshold checking is disabled for batch rating because it decreases pipeline rating performance.

You can enable threshold checking in batch rating by modifying a field in the **multi_bal** instance of the **/config/business_params** object.

To enable credit threshold checking during batch rating:

1. Go to the *BRM_home***/sys/data/config** directory and run the following command:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

   This command creates an editable XML file named **bus_params_multi_bal.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.
2. Search the XML file for the following line:

```
<CreditThresholdChecking>disabled</CreditThresholdChecking>
```

3. Set the **CreditThresholdChecking** entry to the appropriate value:
   - **enabledOffline** specifies to check credit thresholds during the batch rating process.
   - **disabled** specifies to skip credit threshold checking during the batch rating process and thus increase pipeline processing performance. This is the default setting.
4. Go to the *BRM_home***/sys/data/config** directory and run the following command:

   ```
   pin_bus_params bus_params_multi_bal.xml
   ```

   This command loads your changes into the **multi_bal** instance of the **/config/business_params** object.

> **Note:**
>
> BRM uses the XML in this file to overwrite the existing **multi_bal** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM **multi_bal** configuration.

5. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script:

```
pin_multidb -R CONFIG
```

Configuring Batch Controller to Run load_notification_event

Batch Controller is a **cronjob** application that monitors a processing directory and starts batch handlers. When configured for threshold checking, Batch Controller waits for output XML files to appear in the processing directory and then launches the SampleLoadHandler handler. The batch handler moves the output XML files to a separate directory and then calls the **load_notification_event** utility to load the notifications into the BRM database.

You configure Batch Controller by using the following configuration files:

- The **batch_controller.properties** file connects the Batch Controller to the CM, specifies run-time parameters, identifies the batch handlers to run, and specifies when and how to run the batch handlers.
- The **load_notification_event pin.conf** file connects the **load_notification_event** utility to the CM.
- The **SampleLoadHandler_config.values** file specifies where the batch handler retrieves and deposits the XML files and which utility to run.

To configure Batch Controller for threshold checking:

1. Open the *BRM_home***/apps/batch_controller/batch_controller.properties** file in a text editor.
2. Connect Batch Controller to the CM, if you have not already done so.
3. Make sure the entries for the **load_notification_event** batch handler (*BRM_home***/apps/load_notification_event/SampleLoadUtilityHandler.pl**) are correct. For example:

```
SampleHandler.name = SampleHandler
SampleHandler.max.at.lowload.time = 1
SampleHandler.start.string = $PIN_HOME/apps/load_notification_event/
SampleLoadUtilityHandler.pl
```

4. Specify the handler's polling directory and the file pattern to look for. For example:

```
event1.name = eventlabel1
event1.handlers = SampleHandler
event1.at = 115700
event1.file.location = $HOME/ifw/data/out/notifybalancebreach/
event1.file.pattern = *.xml
```

5. Save and close the file.

6. Connect the **load_notification_event** utility to the BRM database by editing the *BRM_home***/apps/load_notification_event/pin.conf** file.
7. Open the *BRM_home***/apps/load_notification_event/SampleLoadHandler_config.values** file in a text editor.
8. Specify the directory from which to retrieve the XML files in the $STAGING entry. For example:

   ```
   $STAGING = "$HOME/ifw/data/out/notifybalancebreach/"
   ```

9. Specify the directory in which the handler archives the output XML files that loaded successfully into BRM in the $ARCHIVE entry. For example:

   ```
   $ARCHIVE = "$HANDLER_DIR/archive"
   ```

10. Specify the directory in which the handler stores the output XML files that failed to load into BRM in the $REJECT entry. For example:

    ```
    $REJECT = "$HANDLER_DIR/reject"
    ```

11. Save and close the file.
12. Start Batch Controller.

Configuring Pipeline Manager to Perform Threshold Checking

To configure threshold checking, perform the following:

1. Configure the DAT_BalanceBatch module.
2. Configure the DAT_PortalConfig module.
3. Configure the FCT_ApplyBalance module.

   When you configure the FCT_ApplyBalance module, you specify the following:

   - The maximum number of notifications in the output file. See "Configuring the Maximum Number of Notifications".
   - The output file prefix and directory location. See "Configuring the Output XML File".

Configuring the Maximum Number of Notifications

You specify the maximum number of notification events that FCT_ApplyBalance writes into the XML output file by using the **NumberOfNotificationLimit** registry entry. When the file breaches the maximum number, FCT_ApplyBalance closes the file and begins writing notification events into a new XML output file.

> **Note:**
>
> You can identify the order in which XML output files were created by the sequence number used in the file name. See "About the Format of the XML Output File Name".

Configuring the Output XML File

After FCT_ApplyBalance identifies that an EDR caused a balance to breach a credit threshold, it writes the EDR to an output XML file. You can specify the prefix for the output file name by using the **OutputPrefix** registry entry and the directory in which to save the file by using the **OutputDirectory** registry entry.

Part V
Suspending and Recycling EDRs

This part describes how to suspend and recycle EDRs in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

43

About the EDR Recycling Features

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) features used to manage suspended (failed) call records (EDRs).

About the EDR Recycling Features

BRM offers these tools for managing EDRs that are not successfully rated by Pipeline Manager:

- **Standard recycling**. BRM provides the standard recycling tools as the default EDR recycling mechanism. Using standard recycling, you use the **pin_recycle** utility to test recycle, recycle, or delete EDRs that failed processing the first time through the pipeline.

  The standard recycling tools include:

  - FCT_Reject
  - FCT_PreSuspense
  - FCT_Suspense
  - **pin_recycle** utility

  For details on standard recycling and configuring the standard recycling tools, see "About Standard Recycling".

- **Suspense Manager**. Suspense Manager is a service integration component that you purchase separately. It offers the most comprehensive and flexible set of tools for managing:

  - Individual failed CDRs
  - Large numbers of individual failed CDRs at once (bulk processing)
  - CDR files containing multiple CDRs (batch processing)

  Suspense Manager includes the Suspense Management Center GUI application to:

  - Analyze, edit, recycle, test recycle, write off, archive, and delete CDRs, either individually or in bulk.
  - Analyze, resubmit, write off, and delete batch files of CDRs.

  Suspense Manager also includes a set of BRM reports for analyzing suspended call records. For details, see "About Suspense Manager".

- **Recycling EDRs for pipeline-only systems**. This feature is used by customers that use Pipeline Manager, but do not store suspended EDRs in the BRM database. This feature includes the FCT_Recycle and FCT_PreRecycle pipeline modules that you use to recycle suspended EDRs. For details see "Recycling EDRs in Pipeline-Only Systems".

44

About Standard Recycling

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) standard recycling features.

For information on the other BRM recycling features, see "About the EDR Recycling Features".

Before using standard recycling, you should be familiar with Pipeline Manager. For details, see "About Pipeline Rating".

About Standard Recycling

You use standard recycling to recycle, test recycle, or delete failed event data records (EDRs).

Standard recycling mainly relies on these BRM tools to suspend and recycle EDRs:

- The "FCT_Reject" pipeline module
- The "FCT_PreSuspense" pipeline module
- The "FCT_Suspense" pipeline module
- The Suspended Event (SE) Loader application
- The "pin_recycle" utility

You use "pin_recycle" to recycle, test recycle, or delete suspended call records. EDRs are often suspended because of a pipeline configuration problem. You then fix the problem, and test recycle a CDR file of suspended call records. If they pass the recycle test, then you recycle all of the CDR files of suspended calls. **pin_recycle** also has a delete option to remove call records that have been successfully processed, or call records that cannot be rated.

Standard Recycling Workflow

Overview of the standard recycling process:

1. You start the pipeline with the **FCT_PreSuspense**, **FCT_Suspense**, and **FCT_Reject** modules active.
2. **FCT_PreSuspense** appends suspense-related information to all EDRs that come through the pipeline.
3. As an EDR is processed, a module finds an error in the EDR. The error is appended to the EDR, and a flag is set to indicate that the EDR has an error.
4. The EDR is sent to the next module. Each module adds errors, if any more are found.
5. The **FCT_Reject** module analyzes an EDR's errors to determine whether it has failed. **FCT_Reject** also routes EDRs to the appropriate output stream to be stored in the database by Suspended Event (SE) Loader. SE Loader stores suspended EDRs in **/suspended_usage** objects

   By default, FCT_Reject fails call records with an error level of **Warning** or **Error**. However, you configure the error level or other conditions that causes EDRs to fail. Call records also "fail" if they cannot otherwise be processed by the pipeline. These failures can be intentional or inadvertent. For example:

- A call record may arrive with invalid data and fail a Pipeline Manager validity rule.
- The call record may fail custom validity checking set up in a custom iScript.
- The Pipeline Manager database tables may be set up incorrectly.

6. During recycling operations, **FCT_Suspense** routes EDRs from **SuspenseCreateOutput** to **SuspenseUpdateOutput**.
7. You examine the errors and determine how to reconfigure Pipeline Manager to prevent the errors.
8. Run the **pin_recycle** utility with the **-f** *filename* option to start the recycling process. This sends the rejected EDRs through the pipeline again for another attempt to rate them.

   pin_recycle can recycle EDRs in test mode or real mode. Typically, you run the recycling processes in test mode first, to see if the problems causing the EDR errors have been fixed. When there are no longer any errors, you recycle in real mode.

   You usually run **pin_recycle** (as part of a **cron** job) periodically.

   - In test mode, this utility creates a report about the processing, but does not make any state changes.
   - In recycle mode, this utility sends the results to an output file, and attaches a sequence number to the output file.

   > **Note:**
   >
   > This utility sends an entire CDR file to the error directory. You can configure the threshold for the number of errors allowed per file. See "Specifying the Maximum Errors Allowed in an Input File".

9. Run **pin_recycle** again with the delete option to remove any remaining unratable EDRs.

For details on the **pin_recycle** utility, see "pin_recycle".

For details on configuring Pipeline Manager to use standard recycling, see "Configuring Standard Recycling".

For details on using standard recycling to recycle, and delete EDRs, see "Using Standard Recycling to Recycle Suspended EDRs".

Suspended EDR States

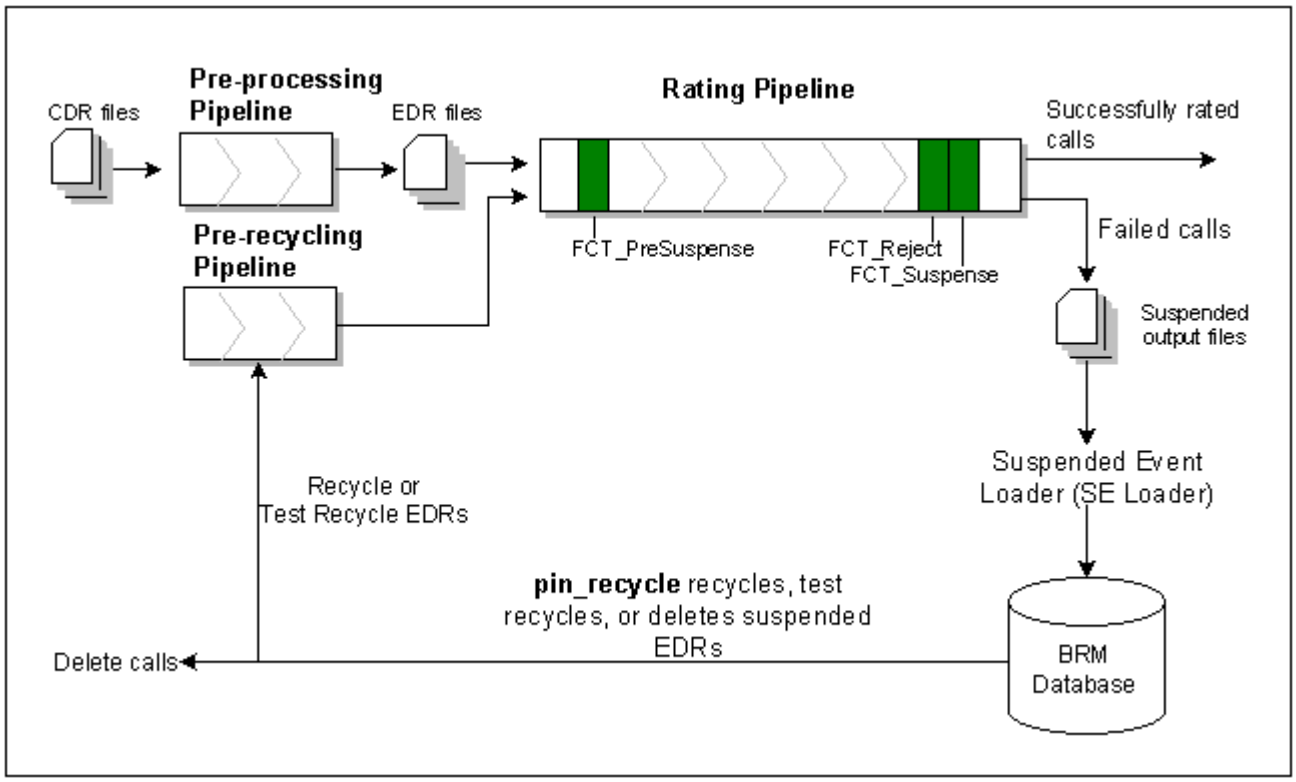
As suspended EDRs are processed by standard recycling, they are assigned one of the following states:

- **Suspended**. The call record could not be processed by the pipeline and has been stored in the BRM database as a suspended call record.
- **Recycling**. The call record is being sent through the rating pipeline again to be rated
- **Succeeded**. The call record has been successfully recycled and rated.
- **Written off**. The EDR is set to this state automatically just before being deleted to generate revenue assurance data.

About the Standard Recycling Pipelines

Figure 44-1 shows how standard recycling fits into your BRM system.

Figure 44-1 Standard Recycling in BRM



Call records first enter standard recycling through the *preprocessing pipeline*. The preprocessing pipeline converts call records (CDRs) to EDRs used by BRM. Calls only go through this pipeline once, so only a few modules are appropriate for it. **FCT_DuplicateCheck** and **FCT_CallAssembling** are candidates.

The *rating pipeline* is a normal rating pipeline. Most of your pipeline function modules are included in this pipeline. It is in this pipeline where you configure call "success" and "failure" policies. If calls "fail" in this pipeline, they are sent to a Suspended Event Loader (SE Loader).

SE Loader converts the failed calls to ***/suspended_usage*** objects in the BRM database.

After these objects are stored in the BRM database, you check your Pipeline Manager log files to see what caused the calls to fail. The problem can frequently be fixed by reconfiguring the pipeline.

Suspended calls that you recycle or test recycle are processed by the *pre-recycle pipeline* before they go through the rating pipeline again. The pre-recycle pipeline converts the suspended call objects back into files that the pipeline can process, and routes the suspended call records back through their original pipeline for recycling.

If you are test recycling calls, the pipeline tries to rate the calls, but does not make any changes to the database.

What's Next

The next step is to configure standard recycling. For details, see "Configuring Standard Recycling".

45

Configuring Standard Recycling

This chapter explains how to set up the Oracle Communications Billing and Revenue Management (BRM) standard recycling feature.

Before you read this document, you should be familiar with:

- Pipeline Manager and how to set up pipeline modules. See these documents:
  - About Pipeline Rating
  - Configuring Pipeline Rating
  - Configuring EDR Input Processing
  - Configuring EDR Output Processing
  - Configuring EDR Preprocessing
- Editing **pin.conf** configuration files and using file loading utilities.

> **Note:**
>
> Suspense Manager customers must complete the configuration instructions in this chapter first, and then follow the instructions in "Installing and Configuring Suspense Manager".

About Configuring Standard Recycling

Table 45-1 lists the tasks required for configuring standard recycling:

Table 45-1 Standard Recycling Configuration Tasks

Task	Description
Configure the pre-processing pipeline See "Configuring a Preprocessing Pipeline".	• Configure your input module. • Configure the OUT_GenericStream pipeline module. • Configure MultiDB routing logic (optional).
Configure the rating pipeline See "Configuring Standard Recycling in a Rating Pipeline".	• Configure the INP_GenericStream pipeline module. • Configure the FCT_PreSuspense pipeline module. • Configure the FCT_Reject pipeline module. • Set RejectStream to SuspenseCreateOutput. • Configure the FCT_Suspense or pipeline module. • Configure the SuspenseCreateOutput registry entry. • Configure the SuspenseUpdateOutput registry entry.

Table 45-1 (Cont.) Standard Recycling Configuration Tasks

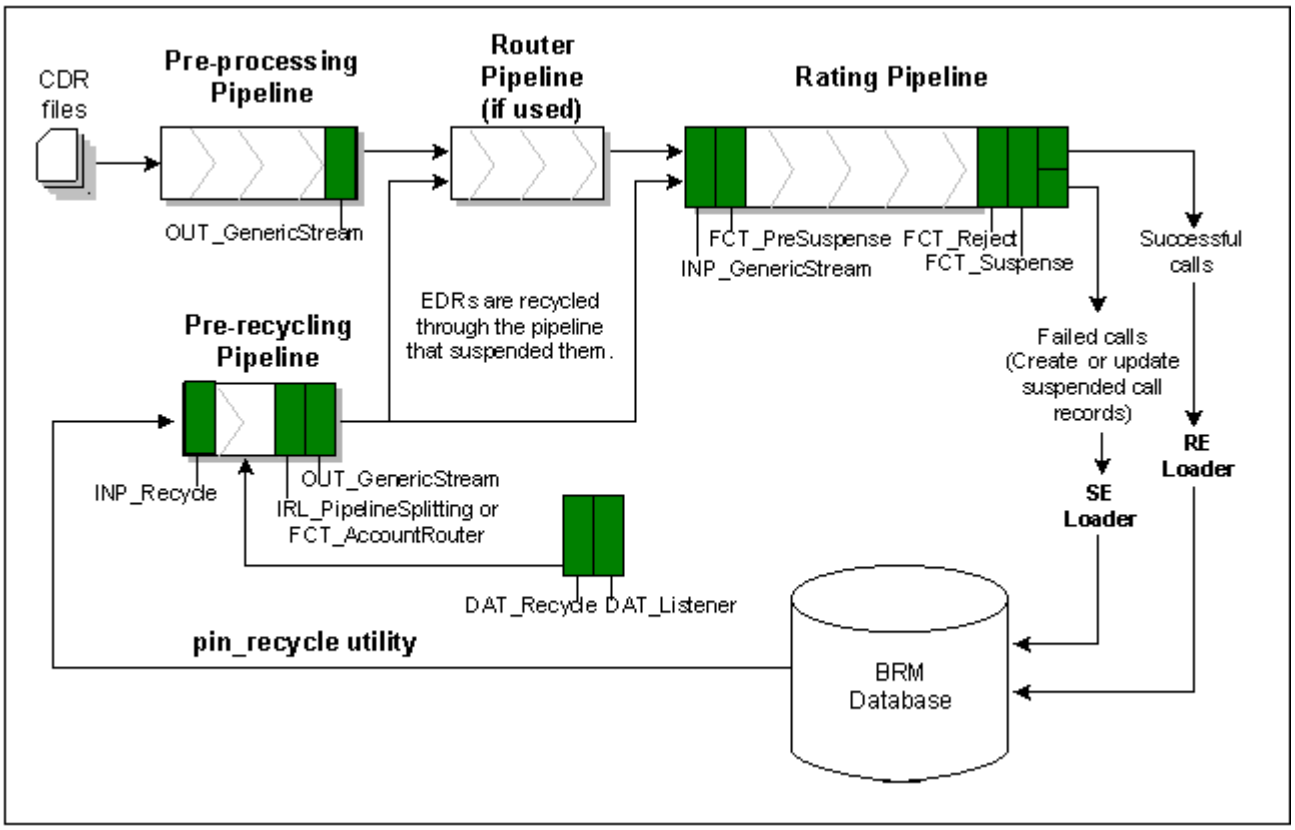
Task	Description
Configure the pre-recycling pipeline See "Configuring a Pre-Recycling Pipeline".	• Configure the INP_Recycle pipeline module. • Edit the IRL_PipelineSplitting.data pipeline module. • Configure the OUT_GenericStream pipeline module.
Configure recycle request handling See "Configuring Recycle Request Handling".	• Configure the DAT_Listener pipeline module. • Configure the DAT_Recycle pipeline module.
Configure a pipeline module to add recycle keys to EDRs (if needed) See "Configuring a Pipeline Module to Add Recycle Keys to EDRs".	• Used by features that temporarily suspend rating.
Configure **pin_recycle** See "Configuring the pin_recycle Utility".	• Set up **pin_recycle** to recycle suspended EDRs.
Set up Suspended Event (SE) Loader See "Configuring SE Loader for Standard Recycling".	• Configure the Batch Controller. • Edit the **Infranet.properties** file.
Confirm that **pin_rel** is configured.	• Ensure that **pin_rel** is configured for standard recycling.

Configuring Pipeline Modules for Standard Recycling

Standard recycling requires you to configure the rating pipeline to correctly handle suspended call records. For an example of a complete sample pipeline, see the *pipeline_home***/conf/wireless.reg**. *pipeline_home* is the directory where you installed Pipeline Manager.

Figure 45-1 shows in green the pipeline modules that you need to configure:

Figure 45-1 Pipeline Modules to Configure for Standard Recycling



Configuring a Preprocessing Pipeline

Standard recycling requires a preprocessing pipeline, and this section explains how to set it up.

For a complete example of a preprocessing pipeline, see *pipeline_home***/conf/wireless.reg**.

All call records coming into your system for rating go through the preprocessing pipeline only once. After the preprocessing pipeline, EDRs go to the rating pipeline. Failed calls may be recycled through the rating pipeline multiple times, but they skip the preprocessing pipeline after going through once.

To configure your preprocessing pipeline:

1. Define preprocessing pipelines in the registry. You need a separate preprocessing pipeline for each input format your system uses.
2. Set up the input module of each pipeline to process call records from the external system you are using. You need a different pipeline for each call record format.
3. Configure the **OUT_GenericStream** pipeline module as an output module of the preprocessing pipeline.
   - Add this entry to the DataDescription.StreamFormats section of each preprocessing pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL.dsc
     ```

   - Add this entry to the DataDescription.OutputMappings section of each preprocessing pipeline:

```
SOL42=./FormatDesc/Solution42/SOL42_V670_REL_OutMap.dsc
```

- Add this entry to the output module section of each preprocessing pipeline:

```
Grammar=./FormatDesc/Solution42/SOL42_V670_REL_OutGrammar.dsc
```

For complete examples of these registries, see *pipeline_home***/conf/wireless.reg**. For details on this module, see "OUT_GenericStream".

4. (Optional) Multidatabase Manager users typically add a Multidatabase Manager routing pipeline after the preprocessing pipeline.

The preprocessing pipeline is now configured. Follow the steps in the next section to configure the rating pipeline.

Configuring Standard Recycling in a Rating Pipeline

All calls go through a rating pipeline at least once, and suspended calls may be recycled through this pipeline multiple times.

For a complete example of a rating pipeline, see *pipeline_home***/conf/wireless.reg**.

> **Note:**
>
> You must use the input description file specified below. Customized description files are not supported.

To configure your rating pipeline:

1. Configure the **INP_GenericStream** pipeline module as the input module.
   - Add this entry to the DataDescription.StreamFormats section of each rating pipeline:

     ```
     SOL42_INPUT=./FormatDesc/Solution42/SOL42_V670_REL_ForInput.dsc
     ```

   - Add this entry to the DataDescription.InputMappings section of each rating pipeline:

     ```
     SOL42_INPUT=./FormatDesc/Solution42/SOL42_V670_REL_InMap.dsc
     ```

   - Add this entry to the input module section of each rating pipeline:

     ```
     Grammar=./FormatDesc/Solution42/SOL42_V670_REL_InGrammar.dsc
     ```

   For examples of these entries, see *pipeline_home***/conf/wireless.reg**. For details on this module, see "INP_GenericStream".
2. Configure **FCT_PreSuspense** as the first function module of the pipeline. For details, see "FCT_PreSuspense".
3. Configure **FCT_Reject** to route suspended calls to the suspense create output module (in Step 5.). For details, see "How the FCT_Reject Module Works" and "FCT_Reject".
4. Set the **RejectStream** entry to **SuspenseCreateOutput** in the rating pipeline:

```
...
ALL_RATE
{
   Active                        = true

   CountryCode                   = 49
   MobileCountryCode             = 262
   NationalAccessCode            = 0
```

```
   InternationalAccessCode     = 00
   InternationalAccessCodeSign = +
   NetworkDestinationCode      = 172
   RejectStream                = SuspenseCreateOutput
...
```

5. Configure **FCT_Suspense** as the last function module of the rating pipeline. You need to configure the registry section of this module. For details, see "FCT_Suspense".
6. Confirm that your pipeline contains a **MaxErrorRates** output entry. If this entry is missing unexpected log file messages may result. For details on adding this entry, see "Specifying the Maximum Errors Allowed in an Input File".
7. Configure the suspense create output module as one of the output modules for this pipeline.

   The following example works as a suspense create output module. Add the **/suspended_usage** object produced by this pipeline in the **EventType** entry:

```
SuspenseCreateOutput

{
   ModuleName                  = OUT_GenericStream

   EventType = /suspended_usage

   Module
   {
      Grammar                   =
./formatDesc/Formats/SuspenseHandling/SuspendedUsageCreationGrammr.dsc
      DeleteEmptyStream         = True

      OutputStream
      {
      ModuleName               = EXT_OutFileManager
      Module
      {
         OutputPath             = ./data/reject
         OutputPrefix           = suspense_create_
         OutputSuffix           = .out
         TempPrefix             = tmp

         TempDataPath           = ./data/reject
         TempDataPrefix         = susp.create.tmp.
         TempDataSuffix         = .data

         Replace                = True
         AppendSequenceNumber   = False
         }
      }
   }
} # end of SuspenseCreateOutput
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of **OutputPath**, **OutputSuffix**, and **OutputPrefix** values for each output stream defined in the registry.

8. Configure the suspense update output module as one of the output modules for this pipeline.

   This example implements a suspense output module:

```
#----------------------------
SuspenseUpdateOutput
{
   ModuleName                  = OUT_GenericStream

   EventType = /tmp_suspended_usage

   Module
   {
      Grammar                   = ./formatDesc/Formats/SuspenseHandling/
SuspendedUsageUpdateGrammar.dsc
      DeleteEmptyStream         = True

      OutputStream
      {
         ModuleName              = EXT_OutFileManager
         Module
         {
         OutputPath            = ./data/reject
         OutputPrefix          = suspense_update_
         OutputSuffix          = .out
         TempPrefix            = tmp

         TempDataPath          = ./data/reject
         TempDataPrefix        = susp.update.tmp.
         TempDataSuffix        = .data

         Replace               = True
         AppendSequenceNumber  = False
         }
      }
   }
} # end of SuspenseUpdateOutput
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

Configuring a Pre-Recycling Pipeline

When suspended call records are recycled, they are first processed by a pre-recycling pipeline and then reprocessed by the original rating pipeline.

The pre-recycling pipeline used the INP_Recycle module. This module is used by standard recycling and Suspense Manager. It reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.

For a complete example of a pre-recycling pipeline, see *pipeline_home***/conf/wireless.reg**.

To configure your pre-recycling pipeline:

1. Configure INP_Recycle as the input module. For details, see "INP_Recycle".

- In the **EXT_InEasyDB** module, change the **SqlDetail** entry to **StdRecycleDetail.sql**.

2. Add and configure a pipeline module to send call records to the correct stream.
   - **Single database schema systems:** Use the "IRL_PipelineSplitting" module. Add and configure the **IRL_PipelineSplitting** module (an iRules module). Add this module to the pipeline registry iRules (its order in the registry is not important). Edit the **IRL_PipelineSplitting.data** file (in the *pipeline_home*/**iScriptLib/iScriptLib_Suspense** directory), adding pipeline name/output stream pairs. For syntax details, see "FCT_IRules".
   - **Multischema systems that require Account Migration Manager:** Use the "FCT_AccountRouter" module. Add and configure this module.

     > **Note:**
     >
     > AMM is not part of base BRM. Contact your BRM account manager for information about using AMM.

3. Configure the **OUT_GenericStream** pipeline module as an output module of the pre-recycling pipeline. Create a different "OUT_GenericStream" module for each rating pipeline used to recycle suspended calls.
   - Add this entry to the DataDescription.StreamFormats section of the pre-recycling pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL.dsc
     ```

   - Add this entry to the DataDescription.OutputMappings section of the pre-recycling pipeline:

     ```
     SOL42=./FormatDesc/Solution42/SOL42_V670_REL_OutMap.dsc
     ```

   - Add this entry to each output module section of the pre-recycling pipeline:

     ```
     Grammar=./FormatDesc/Solution42/SOL42_V670_REL_OutGrammar.dsc
     ```

   For details, see "OUT_GenericStream".

   Your pipelines are now ready to accept call records.

Configuring Recycle Request Handling

To configure recycle request handling, add **DAT_Recycle** to the registry data pool. Here is an example:

```
RecyclingData
{
   ModuleName=DAT_Recycle
   Module
   {
      Listener     =ifw.DataPool.Listener
      LogEvents    =True
      ControlPath  =./database/Oracle/Scripts/Suspense
      ParameterFile =parameter.isc
   }
}
```

For details, see "DAT_Recycle".

Configuring a Pipeline Module to Add Recycle Keys to EDRs

Programs and features that must temporarily interrupt and then restart rating use **pin_recycle** to recycle all EDRs after the interruption is over. The BRM features that add recycle keys to EDRs all have pipeline modules for doing this. See the feature documentation for details.

Configuring the pin_recycle Utility

To configure **pin_recycle** to recycle EDRs, set up a configuration file for **pin_recycle**.

Configuring SE Loader for Standard Recycling

The procedures for installing, configuring, and using SE Loader are identical to those of RE Loader, except for the step listed here. For details see:

- Understanding Rated Event Loader
- Installing Rated Event Loader
- Configuring Rated Event Loader
- Loading Prerated Events

Standard recycling requires SE Loader configuration. Perform these tasks to set it up:

1. Add a separate instance of SE Loader to each pipeline.
2. Create a new *BRM_home***/apps/pin_rel/suspense** directory by copying the contents of *BRM_home***/apps/pin_rel/gsm/tel** to *BRM_home***/apps/pin_rel/suspense**. *BRM_home* is the directory where you installed BRM components.
3. Confirm that these files are in the *BRM_home***/apps/pin_rel/suspense** directory:
   - **pin.conf**
   - **SampleRelHandler_config.values**
   - **SampleRelHandler.pl**
4. Add these entries to the *BRM_home***/apps/pin_rel/suspense/SampleRelHandler_config.values** file:

   ```
   $FILETYPE = "*.out.bc";
   $HANDLER_DIR = "BRM_home/apps/pin_rel/suspense";#
   ```

5. Edit the *BRM_home***/apps/batch_controller/Infranet.properties** file, adding **SUSPENSE** and **RECYCLE_ROLLBACK** entries to **batch.random.events**:

   ```
   batch.random.events     TEL,SMS,FAX,DATA,GPRS,SUSPENSE,RECYCLE_ROLLBACK
   ```

   Add these parameters to the new entries:

   ```
   #for SUSPENSE events:
   SUSPENSE.name                    SUSPENSE Usage
   SUSPENSE.handlers                suspHandler
   SUSPENSE.file.location           pipeline_home/data/reject
   SUSPENSE.file.pattern            suspense_ *.out

   suspHandler.name                 suspHandler
   suspHandler.max.at.highload.time 1
   suspHandler.max.at.lowload.time  1
   suspHandler.start.string           BRM_home/apps/pin_rel/suspense /
   ```

```
SampleRelHandler.pl

#For RECYCLE_ROLLBACK events:
RECYCLE_ROLLBACK.name            RECYCLE_ROLLBACK Usage
RECYCLE_ROLLBACK.handlers        recycleRollbackHandler
RECYCLE_ROLLBACK.file.location   pipeline_home/data/error
RECYCLE_ROLLBACK.file.pattern    testDB*.err

recycleRollbackHandler.name      recycleRollbackHandler
recycleRollbackHandler.max.at.highload.time  1
recycleRollbackHandler.max.at.lowload.time   1
recycleRollbackHandler.start.string      BRM_home/apps/pin_rel/recycle/
SampleRelHandler.pl
```

6. Confirm that these *BRM_home*/**apps/pin_rel/Infranet.properties** file entries are set to **false**:

```
infranet.rel.validate_dbnumber = false
infranet.rel.validate_indexes = false
```

> **Note:**
>
> The SE Loader architecture makes obsolete the database consistency checks and number validation controlled by these entries.

7. Create a new *BRM_home*/**apps/pin_rel/recycle** directory by copying the contents of *BRM_home*/**apps/pin_rel/gsm/tel** to *BRM_home*/**apps/pin_rel/recycle**.
8. Add these entries to the *BRM_home*/**apps/pin_rel/recycle/SampleRelHandler_config.values file**:

```
$FILETYPE = "*.err.bc";
$HANDLER_DIR = "BRM_home/apps/pin_rel/recycle";#
```

9. Add this entry to each output stream of the pre-recycling pipeline in your *pipeline_home*/**conf/wireless.reg** file:

```
EventType = /recycle_suspended_usage
```

46

Using Standard Recycling to Recycle Suspended EDRs

This chapter explains how to use the Oracle Communications Billing and Revenue Management (BRM) **pin_recycle** utility to recycle suspended EDRs. EDRs are usually recycled for one of two reasons:

- They were suspended by the pipeline because of a problem with the pipeline or the EDR. Once the problem is fixed, you recycle the EDRs by using the BRM standard recycling tools in another attempt to rate them. The standard recycling tools recycle all EDRs from the same CDR file at the same time.
- They were suspended intentionally by a BRM program that required a temporary interruption in rating. These programs mark the EDRs with a recycle key and store them until the interruption is over. All EDRs with the same recycle key are recycled at the same time.

In both cases you use **pin_recycle** to recycle the suspended EDRs back through the pipeline to rate them and capture the revenue they represent. See "pin_recycle".

For information on how to configure Pipeline Manager to suspend calls see "Installing and Configuring Suspense Manager".

For details on the **pin_recycle** utility, see "pin_recycle".

About the Standard Recycling Mechanism

The BRM standard recycling feature uses the FCT_Reject, FCT_Suspense, and FCT_PreSuspense pipeline modules, along with the **pin_recycle** utility to suspend and recycle calls that originated in the same CDR input file. After examining the Pipeline Manager log files to determine why calls were suspended, Pipeline Manager administrators fix the pipeline, and then use this utility to attempt to rate these calls again. For details on setting up and using the standard recycling tools, see "Configuring Standard Recycling".

Configuring the pipeline requires system administration experience. You need to be familiar with:

- Modifying BRM pipeline modules to append EDRs with data. For details on setting up and administering pipeline rating, see "About Pipeline Rating".
- Creating a **crontab** file entry to run the **pin_recycle** utility to recycle or delete EDRs. See your operating system documentation for details on creating a **cron** command.

Setting Up EDR Recycling by CDR File

To set up BRM to recycle EDRs by CDR file:

1. Configure the pipeline to reject EDRs according to your business policies. For details, see "Configuring Standard Recycling".

2. Run **pin_recycle** utility with the **-f** option as needed to recycle suspended within a CDR file. You can test recycle, recycle, or delete all the failed EDRs contained in that CDR file. For the complete **pin_recycle** syntax, see "pin_recycle".

   You can run this utility like any other BRM utility, but you will probably want to run it manually as needed. How often you run this script depends on how many EDRs your pipeline rejects. When you make frequent or significant changes to your pipeline, you need to check your log files frequently. If a lot of EDRs are being rejected, you need to run **pin_recycle** often.

About Recycling Suspended EDRs after Rating Interruptions

Some BRM programs and features temporarily interrupt and then restart rating for certain accounts. These programs and features use **pin_recycle** to recycle calls for those accounts when the interruption is over. These features, such as account migration and pipeline-triggered billing, temporarily stop rating by directing the pipeline to suspend calls that come in during the interruption. As these call records arrive in the pipeline, they are appended with a recycle key. When the interruption is over, you use **pin_recycle** to rate all the stored calls that contain that recycle key. You can further configure this feature by using any number of different recycle keys to control when suspended EDRs get recycled.

> **Note:**
>
> This feature is compatible with both Suspense Manager and standard recycling.

The **-k** *recycle key* option directs **pin_recycle** to search for all EDRs that contain a specific recycle key string and a status of **suspended**, and queues them for rating. The BRM feature that suspends EDRs determines which EDRs contain the same recycle key and need to be recycled together. This gives **pin_recycle** the flexibility to selectively restrict recycling to just the EDRs with specific characteristics.

For example, the account migration feature moves groups of accounts across databases, and must temporarily stop rating for each group of accounts while they are being moved. Account migration uses internal job IDs to keep track of the accounts being moved, and it also uses these job IDs in the recycle keys for suspended EDRs associated with those same accounts.

In contrast, the pipeline-triggered billing feature interrupts all rating for all accounts. Therefore, pipeline-triggered billing only needs to use one recycle key (**Trigger_Billing**) for all EDRs that arrive during the temporary suspension.

Before using **pin_recycle**, you must first configure a pipeline module to add the recycle key. For details, see "Setting Up EDR Recycling by Recycle Key".

Setting Up EDR Recycling by Recycle Key

To set up BRM to recycle suspended EDRs by recycle key:

1. Configure the pipeline to suspend EDRs according to your business policies. For details, see "Configuring Standard Recycling".
2. Configure BRM to add the recycle key to EDRs during the temporary interruptions. The feature requiring the temporary interruption has a pipeline module associated with it that does this. For example, the pipeline-triggered billing feature uses the FCT_TriggerBilling module, and Account Migration Manager uses the FCT_AccountRouter module.

> **Note:**
>
> AMM is not part of base BRM. Contact your BRM account manager for information about using AMM.

The recycle key can be any string that corresponds to a set of EDRs to recycle. You configure a pipeline module to add your recycle key to the DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY field of each EDR. The specific module to use depends on the program running billing and the strategy you use for recycling.

3. Configure **pin_recycle** to run periodically. You do this by adding it to a **cron** file. For details, see "Setting Up pin_recycle to Run Periodically".
4. Configure **pin_recycle** to run periodically with the **-d** option to remove the successfully recycled EDRs from the BRM database. You can do this by adding **pin_recycle** to a **cron** file. For details, see "Setting Up pin_recycle to Run Periodically".

Setting Up pin_recycle to Run Periodically

You need to run **pin_recycle** periodically both to queue the temporarily stored EDRs for rating, and to delete them. The **cron** command is the typical way to do this, although you can run **pin_recycle** like any other BRM command-line utility. This section explains how to set up **cron** command to run **pin_recycle**.

You need to add two **pin_recycle** entries to the **cron** command. One to search for and recycle EDRs, and the other to delete them after they are recycled. See "pin_recycle" for the syntax.

Adding EDR Recycle Entries

To run **pin_recycle** periodically, add entries like the following. The optimal frequency depends on your recycling strategy.

Running pin_recycle

Use a **cron** job with a **crontab** entry to run the **pin_recycle** script. The following **crontab** entry runs **pin_recycle** at 1:00 a.m. daily, and queues EDRs with a recycle key of **Trigger_Billing** for rating:

```
0 1 * * * BRM_home/bin/pin_recycle -k Trigger_Billing &
```

BRM_home is the directory where you installed BRM components.

Adding EDR Delete Entries

To remove EDRs from the BRM database, add an entry like the following. The optimal frequency depends on your recycling strategy.

Running pin_recycle

Use a **cron** job with a **crontab** entry to run the **pin_recycle** script. The following **crontab** entry runs **pin_recycle** at 1:00 a.m. daily, and deletes EDRs with a recycle key of **Trigger_Billing**:

```
0 1 * * * BRM_home/bin/pin_recycle -k Trigger_Billing -d &
```

47

About Suspense Manager

This chapter provides an overview of the Oracle Communications Billing and Revenue Management (BRM) Suspense Manager features.

> **Note:**
>
> Suspense Manager is an optional component, not part of base BRM.

Before using Suspense Manager, you should be familiar with the following topics:

- Pipeline Manager. See "About Pipeline Rating".
- Using Pipeline Manager to recycle EDRs. See "Configuring EDR Preprocessing".

> **Note:**
>
> Suspense Manager is an extension of the BRM standard recycling feature. You must configure standard recycling first, before configuring Suspense Manager.

About Suspense Manager

You use Suspense Manager to:

- Analyze, edit, recycle, write off, archive, restore, and delete individual CDRs that have failed pipeline processing.
- Any number of individual call records at the same time (bulk processing).
- Analyze, resubmit, write off, and delete CDR files containing any number of individual call records. CDR files cannot be edited or archived.

> **Note:**
>
> Suspense Management Center and certain BRM utilities and tools refers to CDR files as *batches* or *batch files*.

Suspense Manager includes the Suspense Management Center that allows you to perform these tasks using a graphical user interface.

Records "fail" if they cannot be processed by Pipeline Manager. These failures can be intentional or inadvertent. For example:

- A call record may arrive with invalid data, and fail a pipeline validity rule.
- The Pipeline Manager database tables may be set up incorrectly.

- The call record may fail custom validity checking that you have set up in a custom iScript, such as a size or time duration limit for individual records.

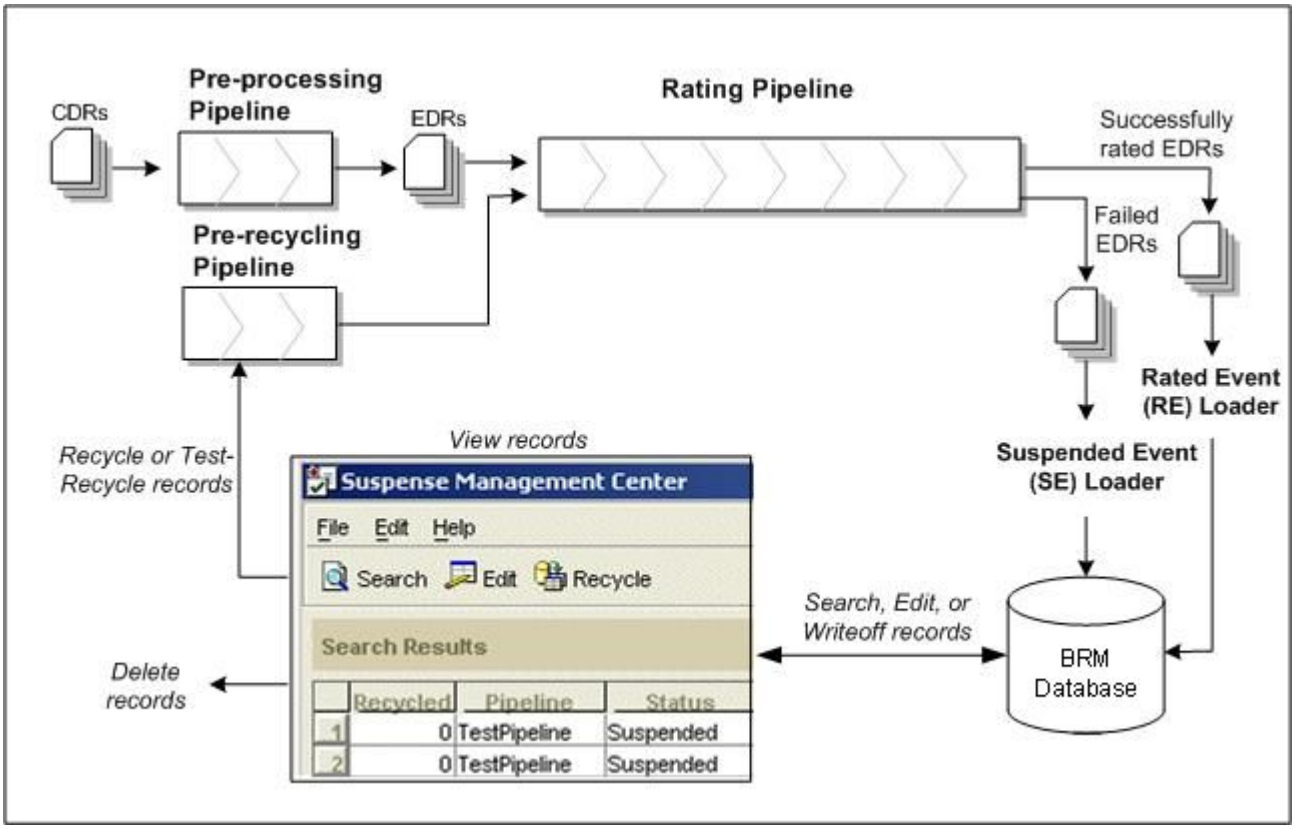
Suspense Manager replaces or augments the base BRM Standard Recycling feature for rejecting or recycling suspended calls.

Suspense Manager server components are available on the Solaris and Linux operating systems and require Oracle database software. The Suspense Management Center client application runs on Windows systems. For details on system requirements, see "Installing and Configuring Suspense Manager".

Suspending Individual CDRs, or CDRs in Bulk

Figure 47-1 shows an example of how Suspense Manager can fit into your BRM system to manipulate individual failed CDRs, on groups of CDRs at once.

Figure 47-1 Suspense Manager and Individual Failed CDRs in BRM



In the example above, CDRs first enter Suspense Manager through a *preprocessing pipeline*. The preprocessing pipeline converts these records to the format used by Suspense Manager, EDRs. These records go through the preprocessing pipeline only once, and only a few modules are needed for it.

This example shows EDRs next going through a normal *rating pipeline*. Most pipeline function modules are included here. CDR "success" and "failure" policies are configured in this pipeline. If records "fail" in this pipeline, they are directed to the appropriate event loader to be loaded into the database.

The *SE Loader* converts the failed calls to objects in the BRM database.

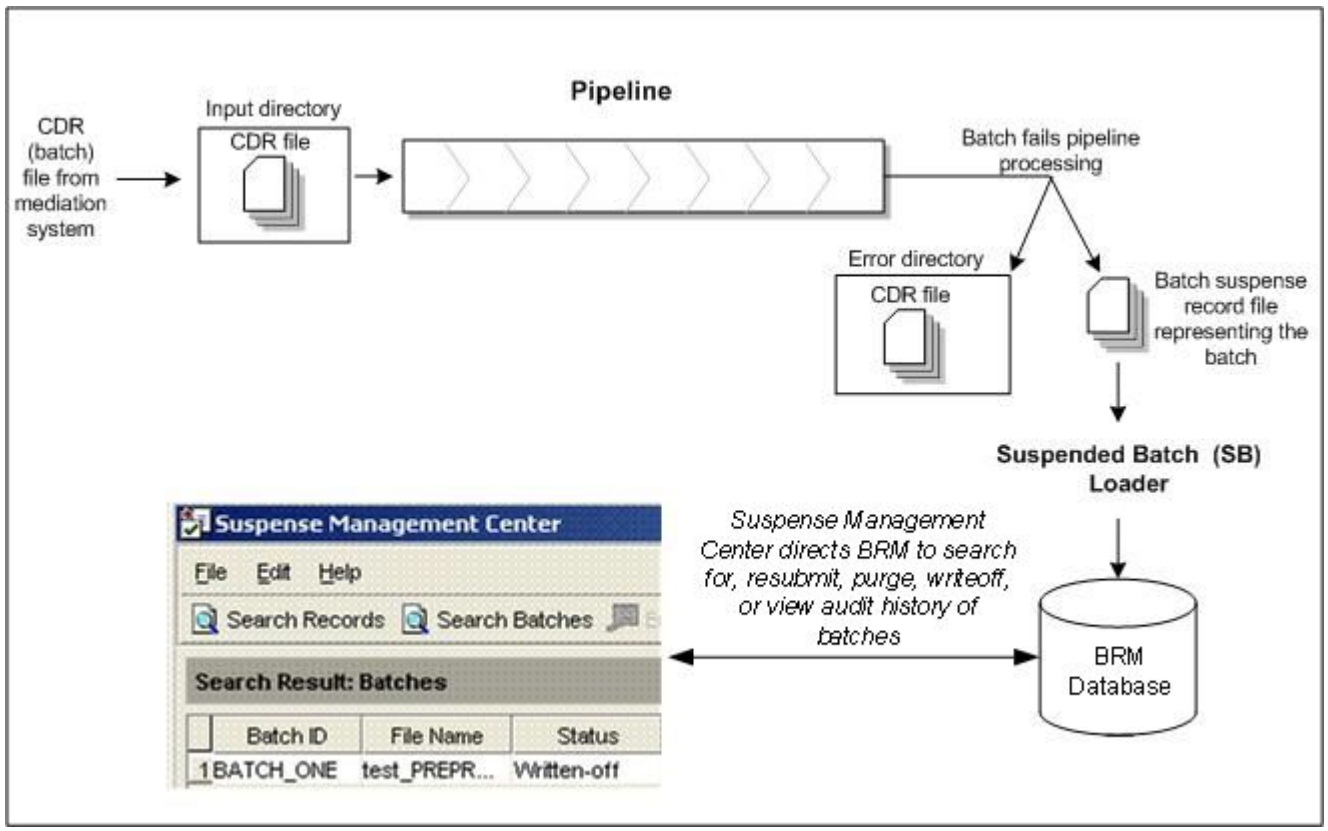
Then, you manipulate these records by using the *Suspense Management Center* application. This application allows you to search for, edit, undo edits, test recycle, recycle, write off, or delete suspended CDRs or CDR files.

In this example, suspended records that get recycled are processed by the *pre-recycle pipeline* before they go through the rating pipeline again. Before recycling these records, you would probably make changes to the rating pipeline, or edit the records so they are successfully rated when they go through the pipeline again. The pre-recycle pipeline converts the suspended record objects back into files that the pipeline can process, and routes the suspended records back through their original pipeline for recycling.

Suspending CDR Files

Figure 47-2 shows an example of how Suspense Manager can fit into your BRM system to manipulate files containing multiple CDRs.

Figure 47-2 Suspense Manager and Files Containing Multiple CDRs



This example shows a CDR file entering through a mediation system. The CDR file is first placed in the pipeline's input directory, and then processed by the pipeline. The pipeline contains "success" and "failure" policies based on file-level or record-level validation. If the CDR file fails pipeline processing, the CDR file itself is directed to the pipeline's error directory, and a **batch_suspense_create** file is created. SB Loader uses the information in this file to create an object which is stored in the BRM database.

You use the Suspense Management Center GUI to manipulate suspended CDR files by acting on the suspended CDR file objects. By using Suspense Management Center, you can resubmit CDR files through the pipeline, purge them, write them off, or view their audit histories.

If the problem with the CDR file is a bad pipeline policy or configuration, you can correct the pipeline and resubmit the CDR file for another attempt at processing.

If the problem is with the CDR file itself, you have several options. You can force the pipeline to ignore certain errors, and process the CDR file, or you can give up on the CDR file and purge it from the database. The list of pipeline errors to ignore is configurable and must be set up ahead of time. You cannot edit CDR files using Suspense Management Center.

Suspended Call Record States

As suspended records are processed by Suspense Manager, they are assigned one of the following states:

- **Suspended**: The call record could not be processed by Pipeline Manager and has been stored in the BRM database as a suspended call record.
- **Recycling**: The call record is being sent through the rating pipeline again to be rated.
- **Succeeded**: The call record has been successfully recycled and rated.
- **Written off**: The call will not be recycled, but is being stored for further use.

Table 47-1 lists the details about the states:

Table 47-1 Suspended Call Record States

State	PIN_FLD_STATUS value in /suspended_usage	Can be edited	Can be recycled	Can be written off	Can be deleted	Can be archived and deleted
Suspended	**0**	Yes	Yes	Yes	No	No
Recycling	**1**	No	No	No	No	No
Succeeded (Successfully processed)	**2**	No	No	No	Yes	Yes
Written off	**3**	No	No	No	Yes	Yes

About Suspended Event (SE) and Suspended Batch (SB) Loader

The Suspended Event (SE) and Suspended Batch (SB) loaders both load suspended records into the BRM database, but they operate on different types of records. The SE loader takes suspended (failed) CDRs as input and uses the **pin_rel** utility to load them into the BRM database as **/suspended_usage** objects. This utility is usually set up to run automatically, but can also be run manually as needed.

The SE loader is a special configuration of the Rated Event Loader (RE Loader), which loads prerated wireless events into the BRM database as objects.

The SB loader does not load CDR files directly into the BRM database. Instead, it accepts information from the **suspense_create_batch** file created for each failed CDR file, and creates **/suspended_batch** objects. The SB loader uses the **load_suspended_batch_info.pl** script to create the **/suspended_batch** objects. This script is usually set up to run automatically, but can also be run manually as needed.

About the FCT_BatchSuspense Module

The FCT_BatchSuspense module adds suspense reason and suspense subreason codes to batches.

- If a resubmitted batch is successful, then FCT_BatchSuspense generates a batch_suspense_update file with status as **Succeeded**. The SB loader reads this file and updates the corresponding /suspended_batch as **Succeeded** when you run **load_suspended_batch_info.pl**.
- If a resubmitted batch fails again, FCT_BatchSuspense generates a batch_suspense_update file with status as **Suspended**, new error code, and new suspense reason. The SB loader reads this file and updates the corresponding / suspended_batch object with a status of **Suspended**.

The specific errors that the FCT_BatchSuspense module adds are based on the error codes assigned to the EDR by the pipeline and the mapping information stored in the /config/ batch_suspense_reason_code object. If no /config/batch_suspense_reason_code object is present, this module sets the suspense reason to **O** (other).

See "FCT_BatchSuspense".

Differences between the RE, SE, and SB Loaders

Table 47-2 explains the differences between the three event loaders:

Table 47-2 DIfferences Between Event Loaders

Task	RE Loader	SE Loader	SB Loader
Loads these types of records	Event file	CDRs	CDR files
Creates these objects	**/event**	**/suspended_usage**	**/suspended_batch**

Suspense Manager APIs

This section briefly describes the Suspense Manager components that you need to customize.

Suspense Manager Objects

Suspense Manager stores individual suspended CDRs in **/suspended_usage** objects, and suspended CDR file records in **/suspended_batch** objects. During configuration, you create a subclass of these objects for each type of call record you receive.

Every action performed by Suspense Management Center is recorded in these **/admin_action** objects:

- **/admin_action/suspended_usage/edit**
- **/admin_action/suspended_usage/recycle**
- **/admin_action/suspended_usage/writeoff**
- **/admin_action/suspended_batch**
- **/admin_action/suspended_batch/resubmit**
- **/admin_action/suspended_batch/writeoff**

For example, when you edit a number of suspended call records at the same time, Suspense Manager records the edits in an **/admin_action/suspended_usage/edit** object. All of the individual suspended CDRs have **/suspended_usage/***type* objects which reference that **/ admin_action/suspended_usage/edit** object.

If you choose to override specific suspense reasons during recycling, the reasons available to override are stored in **/config/suspense_override_reason** objects.

About Upgrading from Standard Recycling to Suspense Manager

To upgrade a system from standard recycling to the Suspense Manager features, follow the instructions in "Installing and Configuring Suspense Manager".

> **Note:**
>
> Suspense Manager is an optional component, not a part of base BRM.

For details on using Suspense Management Center with call records created using standard recycling, see "Using Suspense Management Center with Standard Recycling Call Records".

What's Next

The first step in setting up Suspense Manager is to install the Suspense Manager. software. For details see "Installing and Configuring Suspense Manager".

48

Installing and Configuring Suspense Manager

This chapter explains how to install and set up Oracle Communications Billing and Revenue Management (BRM) Suspense Manager.

Before you read this document, you should be familiar with:

- BRM system administration.
- Pipeline Manager and how to set up pipeline modules. See these documents:
  - About Pipeline Rating
  - Configuring Pipeline Rating
  - Configuring EDR Input Processing
  - Configuring EDR Preprocessing
- Editing **pin.conf** configuration files and using file loading utilities.

Installing Suspense Manager

Before installing Suspense Manager, you must install BRM and then Rated Event (RE) Loader, which is an optional BRM component. For details, see "About Loading Prerated Events".

To install the Suspense Management Center client software, see "Installing Individual BRM Components" in *BRM Installation Guide*.

Installing Suspense Management Center

To install the Suspense Management Center client software, see "Installing Individual BRM Clients" in *BRM Installation Guide* for more information.

Starting and Using Suspense Management Center

To start and use the Suspense Management Center client software, see "Starting a BRM Client Application on Windows" in *BRM Installation Guide* for more information.

About Configuring Suspense Manager

The business decisions you make in the planning phase determine the details of your implementation in the configuration phase.

Table 48-1 shows the tasks required for configuring Suspense Manager:

Table 48-1 Tasks to Configure Suspense Manager

Task	Description
Plan and set up your database. See "Planning and Setting up Your Database for Suspense Manager".	• Decide whether to extend the **/suspended_usage** class. • Select a list of queryable EDR fields. • Add **/suspended_usage** subclasses with queryable fields.
Create a list of editable fields. See "Creating a List of Editable Fields Based on Your /suspended_usage Subclasses".	• Create a list of fields that you want the ability to correct using Suspense Manager.
Load editable fields into the database. See "Loading Editable Fields into the Database".	• Edit the **pin_suspense_editable_flds** file. • Run the **load_pin_suspense_editable_flds** utility.
Change suspense reasons and subreasons (optional). See "Changing the List of Suspense Reasons and Subreasons".	• Decide whether to change the suspense reason lists. • Edit the **suspense_reason_code.en_US** file or the **batch_suspense_reason_code.en_US** file. • Edit the **pin_suspense_reason_code** or **pin_batch_suspense_reason_code** file. • Run the **load_localized_strings** utility. • Run the **load_pin_suspense_reason_code** or **load_pin_batch_suspense_reason_code** utility.
Configure the pipeline for Suspense Manager. See "Configuring Pipeline Manager for Suspense Manager".	• Configure the standard recycling pipeline. • Configure the rating pipeline. • Configure the pre-cycling pipeline.
Configure SE or SB Loader. See "Setting Up Suspended Event (SE) Loader for Suspense Manager". See "Setting Up Suspended Batch (SB) Loader for Suspense Manager ".	• Edit the **Infranet.properties** file.
Create indexes for search templates. See "Creating Indexes for Search Templates".	• Create indexes.
Configure and customize Suspense Management Center. See "Configuring and Customizing Suspense Management Center".	• Add custom fields (edit the **custom.properties** file). • Add custom fields to Web Start (edit **SuspenseManagement_en.jnlp**) (Optional). • Set up permissions for Suspense Management Center.
Configuring event notification for Suspense Manager. See "Configuring Event Notification for Suspense Manager ".	• Consolidate event notification operations.
Configure debugging (optional). See "Configuring Debugging (Optional)".	• Set up Java PCM logging (edit the **Infranet.properties** file). • Edit **RunSM.bat**.

Table 48-1 (Cont.) Tasks to Configure Suspense Manager

Task	Description
Configure the number of records to process in a transaction (optional). See "Configuring the Number of Suspended Records to Process in a Transaction".	• Edit the **pin_suspense_params** file and run **load_suspense_params**.

Planning and Setting up Your Database for Suspense Manager

> **Note:**
>
> The planning process is critical to the successful operation of Suspense Manager. You will be making database schema changes during setup. Changing the database schema *after* you start using Suspense Manager requires a database upgrade that can be time consuming and painful. Be sure to follow the steps in this section carefully.

The process of setting up your database involves these two steps:

- Picking the EDR fields you will use to search for suspended EDRs (queryable fields). If the queryable fields your implementation requires are not in the default BRM objects, you need to create new objects subclasses for them.
- Picking a list of fields that you will allow your personnel to edit. BRM assumes that this list is a subset of the default EDR fields and any new queryable fields. If not, then you will need to create new objects for them.

Once you have decided on these fields to add and/or edit in EDRs, you will then load the list into the BRM database. This will allow suspense Manager to access them.

Deciding Whether You Need to Extend the Suspense Subclasses

Your business will require you to search for and analyze suspended call records (individual CDRs or CDR files). The first step in setting up Suspense Manager is to decide whether the default Suspense Manager storable classes meet your needs. The storable classes you use must contain the fields your business requires to search for the records you need, and analyze their problems. If not, you will need to modify, extend, or replace the default storable classes.

Selecting a List of Queryable EDR Fields

This is a list of EDR fields that you will use to search for and analyze call records. Suspense Management Center allows you to search for suspended calls based on values in these queryable fields, and displays these values in your search results.

> **Note:**
>
> After you specify and load the list of queryable fields into the database, modifying the list involves significant effort. Be sure to plan the list carefully.

Each BRM implementation requires a different list. Making all of your EDR fields queryable degrades performance by using a lot of unnecessary database storage. However, you do need to make enough fields queryable as necessary to meet your business needs.

Start by reviewing the object specifications for the sample Suspense Manager storable classes listed below. If these storable classes contain all the information your business requires, you won't need to make any changes and you can skip the next section. If you need to expand or extend these classes, make a list of the fields you want to make queryable. You will use this list to define custom extensions to **/suspended_usage** object types in the next section.

> **Note:**
>
> You add one set of queryable fields representing one **/suspended_usage** subclass *per pipeline*. For example, for a single pipeline that accepts **/suspended_usage/telco/gsm** records, you can select queryable fields from the **/suspended_usage/telco** and **/suspended_usage/telco/gsm** subclasses. You cannot select queryable fields from **/suspened_usage/telco/gprs**, because it requires a separate pipeline.

/suspended_usage/telco

This storable class stores general wireless call record data in the fields listed in Table 48-2:

Table 48-2 /suspended_usage/telco Storable Class

Field	Description
PIN_FLD_BYTES_IN	Volume of data sent.
PIN_FLD_BYTES_OUT	Volume of data received.
PIN_FLD_CALLED_TO	Phone number being called.
PIN_FLD_CALLING_FROM	Phone number the call originated from.
PIN_FLD_CALL_DURATION	Call time duration.
PIN FLD PRIMARY MSID	Primary MSID.
PIN FLD SERVICE TYPE	Basic service type.
PIN_FLD_START_TIME	Time the call was initiated. **Note**: The start time is not stored in UTC format.
PIN_FLD_USAGE_TYPE	Describes the call scenario, for example, customer-to-customer call, birthday call, closed-user-group call, or friends & family call.

/suspended_usage/telco/gprs

This class stores call record data for generic GPRS (data) calls in the fields listed in Table 48-3:

Table 48-3 /suspended_usage/telco/gprs Storable Class

Field	Description
PIN_FLD_BYTES_IN	Volume of data sent.
PIN_FLD_BYTES_OUT	Volume of data received.
PIN_FLD_CALLED_TO	Phone number being called.
PIN_FLD_CALLING_FROM	Phone number the call originated from.
PIN_FLD_CALL_DURATION	Call time duration.
PIN FLD PRIMARY MSID	Primary MSID.
PIN FLD SERVICE TYPE	Basic service type.
PIN_FLD_START_TIME	Time the call was initiated. **Note**: The start time is not stored in UTC format.
PIN_FLD_USAGE_TYPE	Describes the call scenario, for example, customer-to-customer call, birthday call, closed-user-group call, or friends & family call.
PN_FLD_APN	APN address.
PIN_FLD_GGSN_ADDRESS	GGSN address.
PIN_FLD_NODE_ID	Node ID.
PIN_FLD_SECONDARY_MSID	MSI number.
PIN_FLD_SGSN_ADDRESS	SGSN address.

/suspended_usage/telco/gsm

This class stores call record data for generic GSM (voice) calls in the fields listed in Table 48-4:

Table 48-4 /suspended_usage/telco/gsm Storable Class

Field	Description
PIN_FLD_BYTES_IN	Volume of data sent.
PIN_FLD_BYTES_OUT	Volume of data received.
PIN_FLD_CALLED_TO	Phone number being called.
PIN_FLD_CALLING_FROM	Phone number the call originated from.
PIN_FLD_CALL_DURATION	Call time duration.
PIN FLD PRIMARY MSID	Primary MSID.
PIN FLD SERVICE TYPE	Basic service type.
PIN_FLD_START_TIME	Time the call was initiated. **Note**: The start time is not stored in UTC format.
PIN_FLD_USAGE_TYPE	Describes the call scenario, for example, customer-to-customer call, birthday call, closed-user-group call, or friends & family call.
PIN_FLD_CELL_ID	Network cell ID where the call originated.
PIN_FLD_DESTINATION_SID	Destination MSC or switch ID.
PIN_FLD_DIALED_NUMBER	Number that was called.
PIN_FLD_ORIGIN_SID	Origin MSC or switch ID.

Table 48-4 (Cont.) /suspended_usage/telco/gsm Storable Class

Field	Description
PIN_FLD_SECONDARY_MSID	IMSI field.

All of the fields in subclasses of **/suspended_usage** are queryable.

Adding /suspended_usage Subclasses with Queryable Fields

Suspense Manager uses the **/suspended_usage** storable class to store failed call records in the BRM database. You must extend this class for each type of suspended call record that your business requires. Suspense Manager provides the **/suspended_usage/telco**, **/suspended_usage/telco/gsm**, and **/suspended_usage/telco/gprs** default subclasses to store data for suspended calls.

To add new types of call records to Suspense Manager:

1. Determine how to subclass **/suspended_usage** objects.
2. Create custom **/suspended_usage** objects.

   Use the Storable Class Editor to add a subclass to **/suspended_usage** for each type of call record your business uses.

Creating a List of Editable Fields Based on Your /suspended_usage Subclasses

You use Suspense Management Center to correct these fields in failed EDRs, and then recycle and rate the calls.

Review these objects and create the list of fields you need to change to correct a failed call. All of the fields you added to **/suspended_usage** subclasses are eligible for editing. You can change this list any time by following the steps in "Loading Editable Fields into the Database".

Loading Editable Fields into the Database

To load your list of editable fields into the database for use by Suspense Management Center:

1. Review the list of fields you assembled in "Creating a List of Editable Fields Based on Your /suspended_usage Subclasses".
2. Add these fields to the *BRM_home***/sys/data/config/pin_suspense_editable_flds** file. *BRM_home* is the directory where you installed BRM components.
3. Run the **load_pin_suspense_editable_flds** utility (located in *BRM_home***/bin)** to load the editable fields into the database:

   ```
   load_pin_suspense_editable_flds pin_suspense_editable_flds
   ```

   For details, see "load_pin_suspense_editable_flds".

Changing the List of Suspense Reasons and Subreasons

Suspense Manager adds the specific reasons for call failures to the EDRs it stores. These reasons, called *suspense reasons,* can be divided into more specific *suspense subreasons*. These suspense reasons and subreasons are stored in the call record along with the rest of the call record data. Because they are stored in the call records, you can search for them by using Suspense Management Center. For example, you can search for all the calls that could not be associated with a subscriber.

The Pipeline Manager error messages that actually cause call failures are mapped to these suspense reasons and subreasons. An extensive default error code-to-suspense reason mapping is provided in Suspense Manager. If your business requires different suspense reasons or subreasons, you can change them and their mappings. You can make these changes at any time, but because you may have to upgrade existing data, it is best to have this mapping in place before you go into production.

Deciding whether to Change the Suspense Reason and Subreason Lists

Each suspense reason covers a group of Pipeline Manager error codes. The strings that define these error messages are listed in the **suspense_reason_code.en_US** file (in the *BRM_home***/sys/msgs/suspense_reason_code** directory). These error code strings are mapped to Pipeline Manager error codes in the **pin_suspense_reason_code** file (in the *BRM_home***/sys/data/config** directory).

If you need to change the default mapping or add additional reasons or subreasons, continue with the instructions in "Changing the Suspense Reason and Subreason Lists" that describe the process. If the default suspense reasons and subreasons are appropriate for your business, then skip the rest of this section.

Changing the Suspense Reason and Subreason Lists

If the default error messages or error message mappings do not meet your business needs, follow the steps in this section to change them. You first edit the text file with your new suspense reasons and subreasons, and then load the mapping into the database.

1. Determine your new suspense reasons.

   This is a list of the most common problems that cause calls to fail. It can be as extensive as you like. You can also look at these as *categories* of suspense reasons, because this is the first of two levels. For example, on this level you might use "Validation check failed," then use the next step to add more specific subreasons, such as "Call exceeds maximum time" or "Suspiciously long call."

   > **Note:**
   >
   > Any Pipeline Manager error message without a suspense reason will use the default Pipeline Manager error message.

2. Determine any new suspense subreasons.
3. Edit the *BRM_home***/sys/msgs/suspense_reason_code.en_US** file, adding suspense reasons as strings and mapping them to integers.

   Sample entry for a suspense reason with the ID of 1:

```
STR
      ID = 1 ;
      VERSION = 1 ;
      STRING - "Unable to identify customer information" ;
END
```

Sample entry for a suspense subreason for a suspense reason with the ID of 2:

```
DOMAIN = "suspense_subreason_1" ;
STR
      ID = 2;
      VERSION = 1 ;
      STRING = "B number missing" ;
END
```

> **Note:**
>
> The default string has an ID of 0. This string appears by default in the case of an error that is not mapped to a suspense reason.

> **Note:**
>
> The reason code numbers 65535 and 65534 are reserved for use by BRM.

The format of the **suspense_reason_code.***locale* file is similar to that of the reasons.locale file.

4. Map your suspense reasons to Pipeline Manager error messages by editing the **pin_suspense_reason_code** (for CDRs) or **pin_batch_suspense_reason_code** (for CDR files) file in the *BRM_home***/sys/data/config** directory.

   Excerpt from the default version of **pin_suspense_reason_code**:

```
#ErrorCodes         SuspenseReason       SuspenseSubReason

# Default error (error that is not specified in this file)
00000                    0                    0

# Framework errors
00464                    1                    1
00479                    1                    1
00496                    1                    1
00497                    1                    1
00480                    1                    1
00481                    1                    1
00482                    1                    1

00208                    9                    0
00209                    7                    0
```

5. Load your localized strings into the database by using the **load_localized_strings** utility.

> **Note:**
>
> The **load_localized_strings** utility requires a configuration file to function correctly.

Example syntax:

```
load_localized_strings suspense_reason_code.en_US
```

> **Note:**
>
> If you're loading a localized version of this file, use the correct file extension for your locale.

6. Load your suspense reason code mapping into the database by using the **load_pin_suspense_reason_code** or **load_pin_batch_suspense_reason_code** utility (in the *BRM_home***/bin** directory). For details, see "load_pin_suspense_reason_code" or "load_pin_batch_suspense_reason_code".

> **Note:**
>
> The **load_pin_suspense_reason_code** and **load_pin_batch_suspense_reason_code** utilities require configuration files to function correctly.

Example syntax for CDRs:

```
load_pin_suspense_reason_code pin_suspense_reason_code
```

Example syntax for CDR files:

```
load_pin_batch_suspense_reason_code pin_batch_suspense_reason_code
```

7. Verify that the strings were loaded by displaying the **/strings** objects using the Object Browser or the **robj** command with the **testnap** utility.
8. Stop and restart the Connection Manager (CM).
9. Stop and restart Suspense Management Center.

Your suspense reason and subreason strings are now loaded into the BRM database to be displayed and used by Suspense Management Center.

Configuring Pipeline Manager for Suspense Manager

The following Pipeline Manager configuration steps are required for Suspense Manager.

Configuring a Standard Recycling Pipeline

Follow the instructions in "Configuring Standard Recycling" for your initial Pipeline Manager setup. The Suspense Manager Pipeline Manager configuration builds on the steps used to create a standard recycling pipeline.

Configuring a Rating Pipeline

The following configuration steps are required to configure a rating pipeline.

Configuring FCT_PreSuspense

You added FCT_PreSuspense as the first function module of the pipeline during standard recycling configuration. For details, see "Configuring Standard Recycling".

The FCT_PreSuspense registry requires additional configuration for Suspense Manager. This module requires information from the object class definition you created in "Adding /suspended_usage Subclasses with Queryable Fields".

The example FCT_PreSuspense registry below shows queryable fields for the **/suspended_usage/telco**, **/suspended_usage/telco/gsm**, and **/suspended_usage/telco/gprs** objects.

```
PreSuspense
                {
                  ModuleName                = FCT_PreSuspense
                  Module
                  {
                     Active                  = True
                     QueryableFields
                     {
                       # table name. If more than one table, use a separate block
                       SUSP_USAGE_TELCO_INFO_T
                       {
                          # format : <database_column_name> = <edr_conatiner_field_name>
                          BYTES_IN = DETAIL.VOLUME_RECEIVED
                          BYTES_OUT = DETAIL.VOLUME_SENT
                          CALLED_TO = DETAIL.B_NUMBER
                          #CALLING_FROM = DETAIL.B_NUMBER
                          CALL_DURATION = DETAIL.DURATION
                          PRIMARY_MSID = DETAIL.A_NUMBER
                          SERVICE_TYPE = DETAIL.BASIC_SERVICE
                          START_TIME = DETAIL.CHARGING_START_TIMESTAMP
                          USAGE_TYPE = DETAIL.USAGE_TYPE
                       }
                      SUSP_USAGE_TELCO_GPRS_INFO_T
                      {
                          # format : <database_column_name> = <edr_conatiner_field_name>
                          APN = DETAIL.ASS_GPRS_EXT.APN_ADDRESS
                          GGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.GGSN_ADDRESS
                          NODE_ID = DETAIL.ASS_GPRS_EXT.NODE_ID
                          SECONDARY_MSID = DETAIL.ASS_GPRS_EXT.PORT_NUMBER
                          SGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.SGSN_ADDRESS
                      }
                      SUSP_USAGE_TELCO_GSM_INFO_T
                      {
                         APN = DETAIL.ASS_GSMW_EXT.APN_ADDRESS
                         CELL_ID = DETAIL.ASS_GSMW_EXT.CELL_ID
                          DESTINATION_SID = DETAIL.ASS_GSMW_EXT.TERMINATING_SWITCH_IDENTIFICATION
                         DIALED_NUMBER = DETAIL.ASS_GSMW_EXT.DIALED_DIGITS
                         ORIGIN_SID = DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION
                                      SECONDARY_MSID = DETAIL.ASS_GSMW_EXT.PORT_NUMBER
                      }
                    }
                  }
                }
```

For details on the syntax, see "FCT_PreSuspense".

Configuring SuspenseCreateOutput

Configure the suspense create output module as one of the output modules for this pipeline.

You can use the sample output module in "Configuring Standard Recycling" in a rating pipeline with one change. You need to change the **EventType** entry from **/suspended_usage** to **/suspended_usage/*type***, where *type* is the subclass you created in "Adding /suspended_usage Subclasses with Queryable Fields".

This example shows the **/suspended_usage/telco** being used:

```
SuspenseCreateOutput

{
   ModuleName                  = OUT_GenericStream

   EventType = /suspended_usage/telco

...
```

Configuring a Pre-Recycling Pipeline

You configured a pre-recycling pipeline as part of standard recycling configuration.

The pre-recycling pipeline uses the INP_Recycle module. This module reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.

For Suspense Manager, the INP_Recycle module does the following:

- Sets the process status to either recycling or test recycling, depending on the processing mode selected in Suspense Management Center.
- Sets override reasons in the DETAIL.ASS_SUSPENSE_EDT.OVERRIDE_REASONS field, and the batch ID to DETAIL.ORIGINAL_BATCH_ID.
- Provides feedback to DAT_Recycle about the status of recycling (commit, cancel, or rollback).
- Takes the original batch ID (from a routing switch, mediation system, or other source) from the /suspended_usage object and copies it to DETAIL.ORIGINAL_BATCH_ID. This module also creates a new batch ID for each batch of recycled records, and set it in the HEADER.BATCH_ID and DETAIL.BATCH_ID fields of those records. FCT_PreSuspense appends DETAIL.BATCH_ID with more information to ensure that it remains unique.

To configure a pre-recycling pipeline, see "Configuring a Pre-Recycling Pipeline". These additional steps are also required for Suspense Manager:

1. In the INP_Recycle module, change this EXT_InEasyDB entry:

   ```
   SqlDetail    = StdRecycleDetail.sql
   ```

   to this:

   ```
   SqlDetail    = RecycleDetail.sql
   ```

   This file is used by EXT_InEasyDB for reading queryable fields in **/suspended_usage** objects.

2. Edit the *pipeline_home*/**database/Oracle/Scripts/Suspense/RecycleDetail.sql** script. *pipeline_home* is the directory where you installed Pipeline Manager.
   - Add any custom queryable fields that you added in "Adding /suspended_usage Subclasses with Queryable Fields".
   - Remove any non-editable fields from the SELECT statement to improve performance.

Setting Up Suspended Event (SE) Loader for Suspense Manager

This section explains the configuration steps necessary to load suspended CDRs into the BRM database. You used the steps in "Configuring SE Loader for Standard Recycling" to configure the SE Loader for standard recycling. Follow these steps to configure your **Infranet.properties** file and store your Suspense Manager customizations in **/suspended_usage** objects:

1. Append the contents of **suspense_Infranet.properties** to your **Infranet.properties** file:

```
% cd BRM_home/apps/pin_rel
% cat suspense_Infranet.properties Infranet.properties
```

2. Edit the *BRM_home*/**apps/pin_rel/Infranet.properties** file to create new event types for each of your **/suspended_usage** or **/suspended_batch** subclasses and for temporary objects. Use the **/suspended_usage/telco** section of *BRM_home*/**apps/pin_rel/Infranet.properties** as a guide.

   You use the queryable fields you set up in "Selecting a List of Queryable EDR Fields" in this step.

Setting Up Suspended Batch (SB) Loader for Suspense Manager

This section explains the SB Loader configuration steps necessary to load CDR files into the BRM database.

Follow these steps to configure SB Loader:

1. Add FCT_BatchSuspense as the first functional plugin of the pre-processing pipeline. See "FCT_BatchSuspense".

   > **Note:**
   >
   > This module can be added to any pipeline doing file-level validation, but this is most likely the pre-processing pipeline.

   Specify the object you will use to store suspended CDR file information using the **StorableClass** registry entry in FCT_BatchSuspense. The default object is **/suspended_batch/cdr**. For details, see "FCT_BatchSuspense".

2. Add these entries to your CM's **pin.conf** file to provide connection to the database to store your suspended CDR files:

```
- nap cm_ptr ip_host port_no
- nap login_type 1
- - userid 0.0.0.1 /service/pcm_client 1
- nap login_name root.0.0.0.1
- nap login_pw password
```

Creating Indexes for Search Templates

By default, Suspense Manager does not include any database indexes for searches other than indexes based on POID IDs. You can improve database performance by creating indexes for your most common searches. The example below guides you through the process.

> **Note:**
>
> If there are many indexes on the tables for **/suspended_usage** objects, you run the risk of degrading SE Loader performance during bulk loading of **/suspended_usage** objects. Experiment to find the right balance of indexes for your system.

Example search template:

```
#Suspense Management Template
#Fri Nov 14 09:16:53 PST 2003
PIN_FLD_CALL_DURATION.max=
PIN_FLD_SUSPENSE_REASON.value=<All>
PIN_FLD_CALL_DURATION.selected=false
PIN_FLD_EDITED.value=<All suspended>
PIN_FLD_TEST_SUSPENSE_SUBREASON.value=<All>
PIN_FLD_RECORD_TYPE.selected=true
PIN_FLD_FILENAME.selected=true
PIN_FLD_TEST_ERROR_CODE.min=
PIN_FLD_STATUS.value=Suspended
PIN_FLD_RECORD_TYPE.text=
PIN_FLD_PIPELINE_NAME.text= datadictionary.class=/suspended_usage/telco
PIN_FLD_PIPELINE_ERROR_CODE.max= PIN_FLD_TEST_SUSPENSE_SUBREASON.selected=false
PIN_FLD_SUSPENSE_SUBREASON.selected=false
PIN_FLD_SERVICE_CODE.text=
PIN_FLD_NUM_RECYCLES.max=0
PIN_FLD_PIPELINE_NAME.selected=false
PIN_FLD_SUSPENSE_REASON.selected=true
PIN_FLD_TEST_SUSPENSE_REASON.value=<All>
PIN_FLD_START_TIME.selected=false
PIN_FLD_CALLING_FROM.text=
PIN_FLD_CALL_DURATION.min=
PIN_FLD_NUM_RECYCLES.selected=true
PIN_FLD_FILENAME.text=
PIN_FLD_EDITED.enabled=true
PIN_FLD_CALLED_TO.selected=true
PIN_FLD_STATUS.selected=false
PIN_FLD_TEST_SUSPENSE_REASON.selected=false
PIN_FLD_RECYCLE_T.selected=false
PIN_FLD_TEST_ERROR_CODE.selected=false
PIN_FLD_CALLING_FROM.selected=true
PIN_FLD_CALLED_TO.text=
PIN_FLD_TEST_ERROR_CODE.max=
PIN_FLD_BATCH_ID.selected=false
PIN_FLD_BATCH_ID.text=
PIN_FLD_PIPELINE_ERROR_CODE.min=
PIN_FLD_SERVICE_CODE.selected=false
PIN_FLD_EDITED.selected=false
PIN_FLD_SUSPENSE_SUBREASON.value=<All>
PIN_FLD_NUM_RECYCLES.min=0
PIN_FLD_PIPELINE_ERROR_CODE.selected=true
```

The example search template translates into this SQL statement:

```
SQL> select st.called_to, st.calling_from, s.filename, s.error_code,
SQL> s.suspense_reason, s.num_recycles from suspended_usage_t s,
SQL> susp_usage_telco_info_t st where s.status = 0 and s.num_recycles
SQL> >= 0 and s.num_recycles <= 0 and s.poid_id0 = st.obj_id0;
```

For Oracle databases, use the statements below to determine which indexes would improve performance.

To evaluate this SQL statement, turn on autotrace and run this statement.

This is the output:

```
SQL> set autotrace on;
SQL> select st.called_to, st.calling_from, s.filename, s.error_code,
SQL> s.suspense_reason, s.num_recycles from suspended_usage_t s,
SQL> susp_usage_telco_info_t st where s.status = 0 and s.num_recycles
SQL> >= 0 and s.num_recycles <= 0 and s.poid_id0 = st.obj_id0;
...
13 rows selected.

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (FULL) OF 'SUSP_USAGE_TELCO_INFO_T'
   3    1    TABLE ACCESS (BY INDEX ROWID) OF 'SUSPENDED_USAGE_T'
   4    3    INDEX (UNIQUE SCAN) OF 'I_SUSPENDED_USAGE__ID' (UNIQUE)

Statistics
----------------------------------------------------------
         176  recursive calls
           0  db block gets
          38  consistent gets
           0  physical reads
           0  redo size
        1218  bytes sent via SQL*Net to client
         430  bytes received via SQL*Net from client
           2  SQL*Net roundtrips to/from client
           4  sorts (memory)
           0  sorts (disk)
          13  rows processed
```

The Execution Plan shows a listing of TABLE ACCESS (FULL), indicating that search performance would be better if you had created the indexes. Based on the select statement, add appropriate indexes. In this example add them to both **num_recycles** and the status in **suspended_usage_t**. This sample statement creates those indexes:

```
SQL> create index i_susp_usage_test on suspended_usage_t (status,
SQL> num_recycles);
```

After creating the indexes, rerunning the select statement results in a more efficient Execution Plan:

```
Execution Plan
----------------------------------------------------------
   0        SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (BY INDEX ROWID) OF 'SUSPENDED_USAGE_T'
   3    2       INDEX (RANGE SCAN) OF 'I_SUSP_USAGE_TEST' (NON-UNIQUE)
   4    1     TABLE ACCESS (BY INDEX ROWID) OF 'SUSP_USAGE_TELCO_INFO_T'
   5    4       INDEX (UNIQUE SCAN) OF 'I_SUSP_USAGE_TELCO__ID' (UNIQUE)
```

```
Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          19  consistent gets
          0  physical reads
          0  redo size
          1218  bytes sent via SQL*Net to client
          430  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          13  rows processed
```

The table scan does not read FULL this time, and there are no **recursive calls** and fewer **consistent gets**. The result is a more efficient call.

Configuring and Customizing Suspense Management Center

Follow these procedures to configure and customize the Suspense Management Center application.

Setting Up Permissions for Using Suspense Management Center

Before you can use Suspense Management Center, you must first set up its user permissions in Permissioning Center.

Adding Custom Fields to Suspense Management Center

Edit **custom.properties** in the *BRM_home*/**Program Files/Portal Software/SuspenseManagementCenter/lib** directory, adding any custom fields from your **/suspended_usage** subclasses. Suspense Management Center displays the fields defined in the **custom.properties** files.

This example defines a new field called **Record Type:** to display in Suspense Management Center:

```
#1. Specify the display name:
#
#field.<dd_field_name>.name = <display name>

field.PIN_FLD_RECORD_TYPE.name = Record Type:
```

Adding Custom Fields to Suspense Management Center Web Start

Edit the **SuspenseManager_en.jnlp** file (in the *Web_Start_home* directory), adding any custom fields from your **/suspended_usage** subclasses. The Web version of Suspense Management Center displays the fields that are defined in **SuspenseManager_en.jnlp**.

This example defines a new field called **Record Type:** to display in Suspense Management Center:

```
<resources>
      <j2se version="1.4*"/>
      ...
      ...
      <jar href="lib/Suspense_Management_Help_en.jar"/>
```

```
    <extension name="JavaHelp" href="3plibs/jsoft.jnlp"/>
    <property name="suspensemanagement.home" value="f:/apache/apache2/htdocs" />
    <property name="field.PIN_FLD_RECORD_TYPE.name" value="Record Type:" />
    <property name="field.PIN_FLD_RECORD_TYPE.column" value="Record Type" />
</resources>
```

Configuring Event Notification for Suspense Manager

When suspended EDRs are recycled or written off, Suspense Manager uses event notification to call opcodes that perform the appropriate follow-up operations.

Although any subclass of the /event class can be used to trigger event notification, Suspense Manager generates the following nonpersistent events specifically to use for event notification:

- **/event/notification/suspense/recycle**: By default, when this event occurs, the EAI framework publishing opcode is called.
- **/event/notification/suspense/writeoff**: By default, when this event occurs, PCM_OP_PROCESS_AUDIT_CREATE_WRITEOFF_SUMMARY is called.
- **/event/notification/suspense/delete**: By default, when this event occurs, *no* opcode is called.
- **/event/notification/suspense/edit**: By default, when this event occurs, *no* opcode is called.

Before you can use Suspense Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them.
2. Ensure that the merged file includes entries for these events:
   - (For Revenue Assurance Manager only) From *BRM_home*/**sys/data/config/pin_notify_ra**:

     /event/notification/suspense/writeoff
   - From *BRM_home*/**sys/data/config/pin_notify_ifw_sync**:

     /event/notification/suspense/recycle
3. (Optional) Add entries for these events to your final event notification list:
   - **/event/notification/suspense/edit**
   - **/event/notification/suspense/delete**

   > **Note:**
   >
   > These events are *not* in a default event notification configuration file. You must manually add them to your final event notification list.
4. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list.
5. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger.
6. Load your final event notification list into the BRM database.

Configuring Debugging (Optional)

This section explains how to set up the Suspense Management Center debugging information logs and the Suspense Management Center console. This task is optional.

About Logging Debugging Information

Suspense Management Center provides the following ways for capturing and displaying debugging information:

- The **SuspenseManagementCenter_opcodes.log** log file captures all opcode input and output flists used by Suspense Management Center.
- The **javapcm.log** file contains detailed debugging information. By default, the logging level is set to **0**, the lowest level. The highest level is **3**. You must set the error buffer to **true** to enable **javapcm** logging.
- The **Dloglevel** entry creates a console window for the Suspense Management Center that displays error messages and debugging information.

Setting Up Logging of Debugging Information

1. Set up Java PCM Logging by adding these entries to the **Infranet.properties** file (in the *BRM_home*/**Program Files/Portal Software/SuspenseManagementCenter/lib** directory):

   ```
   infranet.log.level=3
   infranet.log.logallebuf=true
   infranet.log.opcodes.enabled=true
   infranet.log.opcodes.file=SuspenseManagementCenter_opcodes.log
   ```

2. Set up Suspense Management Center console logging by opening the **RunSM.bat** file (in the *BRM_home***Program Files\Portal Software\SuspenseManagementCenter\lib** directory), and changing the **javaw** entry to **java,** and add this parameter:

   ```
   java -Dloglevel="ALL".
   ```

Logging information is now:

- Displayed in the Suspense Management Center console window.
- Available in these files:
  - *BRM_home***Program Files\Portal Software\SuspenseManagementCenter\lib\javapcm.log**
  - *BRM_home***Program Files\Portal Software\SuspenseManagementCenter\lib\SuspenseManagementCenter_opcodes.log**

Configuring the Number of Suspended Records to Process in a Transaction

To configure the number of suspended records you want the suspense management operations to process in a transaction, perform the following tasks:

1. Edit the **pin_suspense_params** file in the *BRM_home*/**sys/data/config** directory to specify the maximum number of records to process in a transaction. The file includes examples and instructions.
2. Load the contents of the file into the **/config/suspense_params** object in the BRM database by using **load_pin_suspense_params**.

   See "load_pin_suspense_params".

Suspense Management Center and the **pin_recycle** utility read the **/config/suspense_params** file to get the number of records to process in each opcode call and determine the number of times to call the opcodes. For more information, see "Processing Suspended Records in Multiple Steps".

Suspense Management Center Permission Types

Table 48-5 shows the permissions you can set for Suspense Management Center.

Table 48-5 Suspense Management Center Permission Types

Permission Type	Provides Permission To . . .	Max Value Applies
/appcenter/suspensemgt	Use Suspense Management Center.	No
/appcenter/suspensemgt/archive_and_purge	Archive and purge call records, save the records to an archive file, and remove them from the BRM database.	Yes
/appcenter/suspensemgt/batch	Search batches.	Yes
/appcenter/suspensemgt/batch_writeoff	Write off batches.	Yes
/appcenter/suspensemgt/batch_purge	Remove batches from the database.	Yes
/appcenter/suspensemgt/batch_resubmit	Resubmit batches and send them back through a pipeline for rating.	Yes
/appcenter/suspensemgt/bulkedit	Edit a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/bulkpurge	Delete a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/bulkrecycle	Recycle a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/bulkwriteoff	Write off a large number of suspended call records in one database operation.	Yes
/appcenter/suspensemgt/edit	Edit suspended call records.	Yes
/appcenter/suspensemgt/purge	Purge call records from the BRM database.	Yes
/appcenter/suspensemgt/records	Search records.	Yes
/appcenter/suspensemgt/recycle	Recycle suspended call records and send them back through a pipeline for rating.	Yes
/appcenter/suspensemgt/restore	Restore call records from an archive file.	No
/appcenter/suspensemgt/undo_edit	Undo edits to suspended call records.	No
/appcenter/suspensemgt/writeoff	Write off suspended call records.	Yes

49

Using Suspense Manager

This chapter provides general information and advice on how to use Oracle Communications Billing and Revenue Management (BRM) Suspense Manager to process your suspended call records.

- For an overview of Suspense Manager and it's capabilities, see "About Suspense Manager".
- For instructions on how to set up and configure Suspense Manager, see "Installing and Configuring Suspense Manager".

Processing a Large Number of Suspended Records

You can define search criteria to edit, delete, recycle, and write off a large number of suspended records. You define the search criteria for a specific action, such as edit, recycle, write off, or delete, in Suspense Management Center. Suspense Manager opcodes then perform the specified action on all the records that meet the search criteria and are in a valid state for the action.

To avoid a large database transaction during bulk operations, you can specify the number of records to process in each transaction in a bulk operation. Based on the number you specify, Suspense Management Center and the **pin_recycle** utility perform several transactions to process the records in the search result. See "Configuring the Number of Suspended Records to Process in a Transaction".

CSRs who perform operations on large numbers of records require specific permissions that allow these operations.

Overriding Pipeline Suspense Handling Rules

During recycling, Suspense Management Center lets you process call records and batched call records that do not pass your pipeline validation rules.

This override feature allows you to capture and temporarily hold suspicious calls in a suspended state until you can inspect them. If they pass inspection, you can override your validation rules and recycle the calls to capture the revenue they represent. The reasons for suspended CDR file are separate from those or individual CDRs and must be handled separately.

You select the override reasons from the Suspense Management Center Recycle screen. The suspense reasons are then overridden for all of the calls in that recycle CDR files. This directs Suspense Manager to successfully process the individual CDRs or CDR files, even though they do not pass your pipeline validation rules.

Changing the List of Override Reasons

The list of override reasons offered to Customer Service Representatives (CSRs) during recycling is configurable. You can change the list at any time by editing the *BRM_home***/sys/data/config/pin_suspense_override_reason** file, and then loading it into your database by

using the **load_pin_suspense_override_reason** utility in the *BRM_home*/**bin** directory. *BRM_home* is the directory where you installed BRM components.

> **Note:**
>
> The **load_pin_suspense_override_reason** utility requires a configuration file to function correctly.

For example:

```
load_pin_suspense_override_reason pin_suspense_override_reason
```

For details on this utility, see "load_pin_suspense_override_reason".

Changing the List of CDR File Override Reasons

The list of CDR file override reasons offered to Customer Service Representatives (CSRs) while resubmitting them is configurable and is separate from the override reasons for individual call records. You can change the list at any time by editing the *BRM_home*/**sys/data/config/pin_batch_suspense_override_reason** file, and then loading it into your database by using the **load_pin_batch_suspense_override_reason** utility in the *BRM_home*/**bin** directory.

> **Note:**
>
> The **load_pin_batch_suspense_override_reason** utility requires a configuration file to function correctly.

For example:

```
load_pin_batch_suspense_override_reason pin_batch_suspense_override_reason
```

Using Suspense Management Center with Standard Recycling Call Records

If you upgraded from standard recycling to Suspense Manager, you will have two types of call records in your database. Call records created using standard recycling use the default **/suspended_usage** fields, and have a suspense reason of **Other**. Records created under Suspense Manager will have a **type** that corresponds to your custom subclasses of **/suspended_usage**, and have the suspense reasons you created during the Suspense Manager installation and configuration process.

The records you created using standard recycling can be recycled, written off, and deleted using Suspense Management Center. To search for *all* records created under standard recycling, search for call records with a:

- **type** of **/suspended_usage**
- Suspense reason of **Other**.

To limit the search further, enter the values for any of the **/suspended_usage** fields used by standard recycling.

Standard Recycling does not produce CDR file suspense records.

Troubleshooting Suspense Manager

This section contains fixes to common Suspense Manager problems.

Increasing Heap Size to Avoid Performance Problems

If the searches you run in Suspense Management Center return particularly large results, your performance may slow noticeably, or you may get "Out of memory" error messages. The solution is to increase your maximum heap size. The exact amount varies greatly with your needs and system resources. If performance is very bad or you get "Out of memory" messages frequently, start by doubling the maximum heap size to 128 MB. Remember, however, that making the heap size too large will degrade the performance of other processes.

There are two ways to increase the maximum heap size, depending on whether you have standalone or WebStart BRM implementations.

Increasing Heap Size for Standalone Implementations

1. Edit the *BRM_home_dir*/**lib/runSMC.bat** file to increase the heap size (memory allocation pool) to solve "Out of memory" messages.

   By default, Suspense Management Center has a maximum heap size of 64 MB. This variable is controlled by the **-Xmx** *size* entry in the Suspense Manager Center startup line in **runSMC.bat**. No **-Xmx** *size* entry is present in the startup line by default. To increase the heap size, add this entry and a number (in megabytes) to the Suspense Management Center startup line.

   This example adds a 128 MB maximum heap size to Suspense Management Center:

```
@start C:\PROGRA~1\COMMON~1\PORTAL~1\JRE\bin\javaw.exe -Xmx128m -cp ".;%SMCDIR%;%SMCDIR%\lib;%SMCDIR%
\lib\suspensemgtmain.jar;%SMCDIR%\lib\pfc.jar;%SMCDIR%\3plibs\jh.jar;%SMCDIR%\lib\pcmext.jar;%SMCDIR%
\lib\pcm.jar;%SMCDIR%\lib\Suspense_Management_Help_en.jar;%SMCDIR%
\lib\Application_Center_Help_en.jar;" com.portal.appcenter.AppCenterMain suspensemgtsuite
```

   > **Note:**
   >
   > Be sure to precede and follow the **-Xmx** *size* entry with a space.

2. Stop and restart Suspense Management Center to make the change take effect.

Increasing Heap Size for Web Start Implementations

1. Open your **SuspenseManagement**_*locale*.**jnlp** file.
2. Change the **j2se** element to include a max-heap-size attribute.

   The default entry looks like this:

```
<j2se version="1.4*"/>
```

   For example, this entry changes the maximum heap size to 128 megabytes:

```
<j2se version="1.4*" max-heap-size="128m"/>
```

> **Note:**
>
> The max heap size specified in the JNLP file is used for all associated Suspense Management Center clients.

3. Stop and restart Suspense Management Center to make the change take effect.

Unexpected Log Messaged Caused by Missing MaxErrorRates Entry

Your pipeline output section must contain a MaxErrorRates module containing at least one entry. If this entry is missing, your log files will contain a misleading message like this one:

```
"16.11.2004 21:00:37 All checks are successful. File can be recycled."
```

Suspense Manager Performance

The more call records you attempt to edit, recycle, delete, archive, or archive then delete, the longer it takes. It is impossible to say exactly how long because every implementation is different, but 30,000 records will take a few minutes, and recycling 300,000 records will take many minutes.

50
Suspense Reasons

This chapter describes the default suspense reasons in Oracle Communications Billing and Revenue Management (BRM).

Table 50-1 shows the default mapping between various BRM error messages and Suspense Manager reasons for suspending event data records (EDRs). The information in this table is derived from several different source files and is much easier to understand in this format.

You may want to change this mapping or add your own error messages as appropriate for your BRM implementation. For information on how to add to or change these messages, see "Changing the List of Suspense Reasons and Subreasons".

Table 50-1 Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
Pipeline Framework Errors	NA	NA
ERR_INPUT_MAPPING_FAILED	Batch rating engine processing error	Record mapping error
ERR_TRANSFER_CUTOFF_VIOLATED	Batch rating engine processing error	Record mapping error
ERR_INCORRECT_FILLER_LENGTH	Batch rating engine processing error	Record mapping error
ERR_CANNOT_JOIN_EVENT_HANDLER_PROC	Batch rating engine processing error	Record mapping error
ERR_INVALID_RECORD_NUMBER	Batch rating engine processing error	Record mapping error
ERR_INVALID_FIRST_CALL_TIMESTAMP	Batch rating engine processing error	Record mapping error
ERR_INVALID_LAST_CALL_TIMESTAMP	Batch rating engine processing error	Record mapping error
ERR_A_CUSTOMER_NOT_FOUND	Customer data error	Customer error
ERR_NO_SPLITTING_PERFORMED	Splitting error	Splitting error
ERR_ADD_DATABLOCK	Batch rating engine processing error	Batch rating engine processing error
ERR_DATABASE	Batch rating engine processing error	Database error
ERR_RATEPLAN_NOT_FOUND	Rating error	Invalid charge
ERR_INSUFFICIENT_MEMORY	Batch rating engine processing error	Insufficient memory
DAT_Account Errors	NA	NA
ERR_BALANCE_GROUP_UPDATE	Customer data error	Internal error
ERR_BILL_INFO_UPDATE	Customer data error	Internal error
ERR_MAPPING_TABLE_UPDATE	Customer data error	Internal error

Table 50-1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
ERR_CUSTOMER_LOGIN_NOT_FOUND	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_SERVICE_NOT_FOUND	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_ACCOUNT_NOT_FOUND	Customer data error	Customer error
ERR_SERVICE_NOT_CONFIGURED	Customer data error	Customer error
ERR_CUSTOMER_SERVICE_NOT_FOUND	Customer data error	Customer error
ERR_CUSTOMER_EDR_PARSING	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_NOT_VALID_FOR_TIME	Customer data error	Customer error
ERR_CUSTOMER_NO_VALID_PRODUCT	Customer data error	Customer error
ERR_CUSTOMER_NO_VALID_PRODUCT_RATING	Customer data error	Customer error
ERR_CUSTOMER_INVALID_ITEM_POID	Customer data error	Customer error
ERR_INVALID_OUTPUT_STREAM	Customer data error	Customer error
ERR_CUSTOMER_LOGIN_INTERNAL_ERROR	Customer data error	Internal error
ERR_ACCRT_MESSAGE	Customer data error	Internal error
ERR_NOT_IMPLEMENTED	Customer data error	Internal error
FCT_AggreGate Errors	NA	NA
ERR_NO_DEPENDENT_CLASS_DEFINED	Record aggregation error	Aggregation error
ERR_EDR_ITERATOR_FAILURE	Record aggregation error	Aggregation error
FCT_BalancePlugIn Errors	NA	NA
ERR_BALANCE_NOT_FOUND	Billing record error	Failed to add discount balance info
FCT_CallAssembling Errors	NA	NA
ERR_CHAIN_REFERENCE_MISSING	Call assembling error	Call assembling error
ERR_INVALID_STATE_INDICATOR	Call assembling error	Call assembling error
ERR_REJECTED_EDR_NOT_IN_WORKFILE	Call assembling error	Call assembling error
ERR_EDR_ALREADY_CLOSED	Call assembling error	Call assembling error
FCT_CustomerRating Errors	NA	NA
ERR_CUSTOMER_NOT_FOUND	Rating error	Customer rating error
ERR_RATEPLAN_NOT_DEFINED	Rating error	Customer rating error
FCT_Discount Errors	NA	NA
ERR_INVALID_GRANT_TYPE	Discount processing error	Discount processing error
ERR_PLUGIN_INVALID_STATE	Discount processing error	Discount processing error
ERR_EDR_ITERATOR	Discount processing error	Discount processing error
ERR_EDRPACK_NOT_READY_DSC	Discount processing error	Discount processing error
ERR_BEGIN_EDR	Discount processing error	Discount processing error
ERR_COMMIT_EDR	Discount processing error	Discount processing error

Table 50-1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
ERR_CANCEL_EDR	Discount processing error	Discount processing error
ERR_ROLLBACK_EDR	Discount processing error	Discount processing error
ERR_CANCEL_DEMANDED_EDR	Discount processing error	Discount processing error
ERR_BEGIN_DSC_TRANSACTION	Discount processing error	Discount processing error
ERR_END_DSC_TRANSACTION	Discount processing error	Discount processing error
ERR_DSC_CONF_NOT_FOUND	Discount processing error	Discount processing error
ERR_INVALID_THRESHOLD_TYPE	Discount processing error	Discount processing error
ERR_INVALID_DISCOUNT_TYPE	Discount processing error	Discount processing error
ERR_DISCOUNT_DETACH_MODULE	Discount processing error	Discount processing error
ERR_ACCOUNT_COMMIT_RESTART	Discount processing error	Discount processing error
ERR_ACCOUNT_PREPARECOMMIT	Discount processing error	Discount processing error
ERR_ACCOUNT_COMMIT	Discount processing error	Discount processing error
ERR_ACCOUNT_CANCEL	Discount processing error	Discount processing error
ERR_ACCOUNT_ROLLBACK	Discount processing error	Discount processing error
ERR_NO_ACCOUNT_BALANCE_ERROR	Discount processing error	Discount processing error
ERR_CANNOT_COMPILE_SCRIPT	Discount processing error	Discount processing error
ERR_CURRENCY_RESID_NOT_FOUND	Discount processing error	Discount processing error
ERR_INVALID_BASE_AMOUNT	Discount processing error	Discount processing error
ERR_INVALID_BASE_EXPR	Discount processing error	Discount processing error
ERR_INVALID_COND_AMOUNT	Discount processing error	Discount processing error
WRN_INVALID_DRUM_AMOUNT	Discount processing error	Discount processing error
ERR_INVALID_THRESHOLD_AMOUNT	Discount processing error	Discount processing error
ERR_REJECT_EDR	Discount processing error	Discount processing error
ERR_EXPR_REF_CP	Discount processing error	Discount processing error
ERR_GETTING_BG_ID	Discount processing error	Discount processing error
FCT_EnhancedSplitting Errors	NA	NA
ERR_NO_SPLITTING_ENTRY	Splitting error	Splitting error
FCT_ExchangeRate Errors	NA	NA
ERR_EXCHANGERATE_BRK_HEADERDATE	Rating error	Exchange rate error
ERR_EXCHANGERATE_FILEDATE_NOT_EXIST	Rating error	Exchange rate error
ERR_EXCHANGERATE_NOT_FOUND	Rating error	Exchange rate error
FCT_MainRating Errors	NA	NA
ERR_RATEPLAN_VERSION_ID_NOT_FOUND	Rating error	Invalid charge
ERR_RATEPLAN_VERSION_DATE_NOT_FOUND	Rating error	Invalid charge
ERR_TIMEMODEL_NOT_FOUND	Rating error	Invalid time model or time zone
ERR_RATE_PRICEMODEL_NOT_FOUND	Rating error	Invalid pricing

Table 50-1 (Cont.) Suspense Reasons

BRM Error Messages	Suspense Reason String	Suspense Subreason String
ERR_TIMEZONE_NOT_FOUND	Rating error	Invalid time model or time zone
ERR_PRICEMODEL_NOT_FOUND	Rating error	Invalid pricing
ERR_PRICEMODEL_RUM_NOT_FOUND	Rating error	Invalid pricing
ERR_PRICEMODEL_STEP_NOT_FOUND	Rating error	Invalid pricing
ERR_PRICEMODEL_CONFIG_NOT_FOUND	Rating error	Invalid pricing
ERR_INVALID_ADDON_TYPE	Rating error	Other main rating error
ERR_RUM_GROUP_NOT_FOUND	Rating error	Other main rating error
FCT_PreRating Errors	NA	NA
ERR_RATEPLAN_NOT_A_NUMBER	Rating error	Invalid charge
ERR_RATEPLAN_TYPE_INV	Rating error	Invalid charge
ERR_RATEPLAN_VERSION_NOT_FOUND	Rating error	Invalid charge
ERR_NO_RATEPLAN	Rating error	Invalid charge
FCT_Tadig2PlmnPlugIn Errors	NA	NA
ERR_TADIG_NOT_FOUND	Mapping problem	TADIG to PLMN map error
Ciber Errors	NA	NA
ERR_CIBER_RET	Roaming records error	CIBER record error
Tap Errors	NA	NA
ERR_TAP3_RET	Roaming records error	TAP record error
FCT_AccountRouter Errors	NA	NA
ERR_JOB_AMT_MIGRATION	Internal suspension	Account being migrated
FCT_TriggerBill Errors	NA	NA
ERR_TRIGGER_BILLING	Internal suspension	Awaiting billing of account
FCT_Account Errors	NA	NA
ERR_JOB_RERATING_ACCOUNT	Internal Suspension	Account usage being re-rated
FCT_Filter_Set Errors	NA	NA
ERR_INVALID_DISCOUNT_ID	Filter set error	Invalid Discount ID
ERR_INVALID_PRODUCT_ID	Filter set error	Invalid Product ID

51

About Suspense Manager Opcodes

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Suspense Manager standard opcodes. These opcodes manage suspended EDRs stored in the BRM database as **/suspended_usage** objects. These opcodes also manage the Batch Suspense Record of suspended CDR file, stored in the BRM database as **/suspended_batch** objects.

For information about Suspense Manager, see "About Suspense Manager".

Recycling Suspended EDRs

PCM_OP_SUSPENSE_SEARCH_RECYCLE searches for and queues suspended call records for recycling. There are many search criteria that you may use to search the records, such as by CDR file or by recycle key.

This opcode is usually called by the **pin_recycle** utility, which searches for and deletes call records with a specific recycle key and a status of **Succeeded** (or **Written-off**).

Searching for EDRs in a CDR File

The BRM standard recycling feature collects and manipulates all calls contained in a CDR file at the same time. PCM_OP_SUSPENSE_SEARCH_RECYCLE searches for a specified CDR file and queues its suspended EDRs for recycling. If successful, the EDRs are assigned a status of **Succeeded**.

Searching for EDRs with a Recycle Key

PCM_OP_SUSPENSE_SEARCH_RECYCLE is used by features that need to temporarily delay rating of EDRs. These features include pipeline modules to add a recycle key and error to those EDRs during pipeline rating. This opcode searches for all **Suspended** call records containing the recycle key passed in on the input flist. It then queues those call records to be rated at the next opportunity. If successful, the EDRs are assigned a status of **Succeeded**.

Initiating Suspense Recycling

PCM_OP_SUSPENSE_RECYCLE_USAGE initiates EDR recycling. During recycling, suspended EDRs are sent back through their original rating pipelines. The Suspense Management Center calls this opcode when the user chooses to recycle suspended EDRs.

PCM_OP_SUSPENSE_RECYCLE_USAGE can operate in test mode. In test mode, the EDR is sent through the rating pipeline normally. The rating results, however, are not output from the pipeline if the EDR is processed successfully. In test mode, the pipeline does not make any state changes, such as updating aggregation counters in discounting.

PCM_OP_SUSPENSE_RECYCLE_USAGE takes as input an array of **/suspended_usage** object POIDs, a list of suspense override reasons, and a value that indicates whether the EDRs should be recycled in test mode. This opcode then creates an **/admin_action/suspended_usage/recycle** object with that information.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_RECYCLE_USAGE performs these operations:

- Confirms that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
- Changes the EDR status to **Recycling**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is set to **1**.
- Creates an **/admin_action/suspended_usage/recycle** object containing the recycle mode and override reasons (passed in the input flist).
- Adds the POID of the newly created **/admin_action/suspended_usage/recycle** object to the actions array of each **/suspended_usage** object.
- Sets the PIN_FLD_RECYCLE_OBJ field of each **/suspended_usage** object to the **/admin_action/suspended_usage/recycle** object POID. Pipeline Manager uses this field to find all EDRs associated with a recycle request.
- Creates a notification event that triggers the Account Synchronization Data Manager (**dm_ifw_sync**) to send a message to the DAT_Listener module. This initiates the process of recycling the EDR through the rating pipeline.

After EDRs are recycled, their **/suspended_usage** objects are updated by using the Suspended Event (SE) Loader:

- If an EDR is successfully recycled, the status is changed to **Succeeded**, and the PIN_FLD_STATUS value is changed to **2**.
- If an EDR is not successfully recycled, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. In addition, the following fields are updated because their original values could have changed:
  - The suspense reason code, PIN_FLD_SUSPENSE_REASON.
  - The suspense subreason code, PIN_FLD_SUSPENSE_SUBREASON.
  - The Pipeline Manager error code, PIN_FLD_PIPELINE_ERROR_CODE.
- If PCM_OP_SUSPENSE_RECYCLE_USAGE was run in test mode, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. The suspense reason Pipeline Manager error code are not updated, but the corresponding test values (PIN_FLD_TEST_SUSPENSE_REASON, PIN_FLD_TEST_SUSPENSE_SUBREASON, and PIN_FLD_TEST_SUSPENSE_ERROR_CODE) are updated to include information about the results of the test recycling.
- The number of times the record has been recycled (in PIN_FLD_NUM_RECYCLES) is increased by 1.

PCM_OP_SUSPENSE_RECYCLE_USAGE returns the routing POID specified in the input flist.

Resubmitting Suspended Batches

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES resubmits suspended CDR file. When resubmitted, suspended CDR file are sent back through their original pipeline. Suspense Management Center calls this opcode when the user chooses to resubmit suspended batches.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES takes an array of **/suspended_batch** object POIDs and a list of suspense override reasons as input. This opcode then creates an **/admin_action/suspended_batch/resubmit** object with that information.

For the whole set of **/suspended_batch** objects specified in the input flist, PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES performs these operations:

- Creates a transaction if it is not already opened.
- Creates an ADMIN_ACTION object, **/admin_action/suspended_batch/resubmit**, with the override reason, for each **/suspended_batch** object.
- Validates the status of each **/suspended_batch** object (Batch Suspense Record) and updates the status with the result of the resubmission.
- Creates an event Flist (with **/event/notification/suspense/batch_resubmit**) and calls PCM_OP_ACT_USAGE in CALC_ONLY mode.
- Closes the transaction

After CDR file are resubmitted, their **/suspended_batch** objects (Batch Suspense Records) are updated by using the SE Loader:

- The PIN_FLD_NUM_RESUBMISSIONS field in each **/suspended_batch** object is incriminated.
- If a batch is successfully resubmitted, the status is changed to **Succeeded**, and the PIN_FLD_STATUS value is changed to **2**.
- If a batch is not recycled successfully, the status is changed back to **Suspended**, and the PIN_FLD_STATUS value is changed to **0**. This will cause all the batches in the resubmission task to be rolled back as well. In addition, these fields are updated because their original values could have changed:
  - The suspense reason code, PIN_FLD_BATCH_SUSPENSE_REASON.
  - The suspense sub-reason code, PIN_FLD_BATCH_SUSPENSE_SUBREASON.
  - The Pipeline Manager error code, PIN_FLD_PIPELINE_ERROR_CODE.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES returns the routing POID specified in the input flist.

Changing the Contents of Fields in Suspended EDRs

Use PCM_OP_SUSPENSE_EDIT_USAGE to change the contents of fields in suspended EDRs. Suspense Management Center calls this opcode to edit a suspended call record.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

> **Note:**
>
> You cannot edit the CDR or EDR fields of records in a CDR file that has been suspended by batch suspense and has a Batch Suspense Record.

PCM_OP_SUSPENSE_EDIT_USAGE performs these operations:

- Takes as input an array of **/suspended_usage** object POIDs and an array of the EDR fields to be edited, including the old and new values.

- For each EDR field to be edited, this opcode:
  - Creates an **/admin_action/suspended_usage/edit** object, which includes both the old and new values.
  - Adds the **/admin_action/suspended_usage/edit** object POID to the top of the **/suspended_usage_edits** object stack. If the stack is full, it removes the oldest POID.
- For each **/suspended_usage** object, this opcode:
  - Confirms that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
  - Adds the **/admin_action/suspended_usage/edit** object POID and the old value to each **/suspended_usage** object's action array.
  - Updates the modified flag (PIN_FLD_EDITED), which indicates that the field has been edited.

PCM_OP_SUSPENSE_EDIT_USAGE returns an array of the POID of the **/admin_action/suspended_usage/edit** objects it creates.

Undoing Edits to Suspended EDRs

Use the PCM_OP_SUSPENSE_UNDO_EDIT_USAGE opcode to undo edits to suspended EDRs.

This opcode is called by Suspense Management Center to the undo edits. It replaces the value of a field in a suspended call record with the value in that field before the last edit was made.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE performs these operations:

- Confirms that the POID of the **/admin_action/suspended_usage/edit** object on the input flist matches that of the top POID of the **/suspended_usage_edits** object stack. If the two don't match, returns failure status (PIN_FLD_RESULT is set to **1**) and returns the POID at the top of the **/suspended_usage_edits** object stack.
- Confirms that each suspended call record affected by the edit has a status of **Suspended**.
- Undoes the edit by replacing the existing field values with the values before the last edit was saved (the values referenced by the POID at the top of the **/suspended_usage_edits** object stack).
- Adds the session, service, and date and time details of the undo edit operation in the PIN_FLD_UNDO_DATA array of the **/admin_action/suspended_usage/edit** object.
- Removes the POID of the **/admin_action/suspended_usage/edit** object from the top of the **/suspended_usage_edits** object stack.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE returns a **PIN_FLD_RESULT** value of:

- **0** if the undo edit was successful. Also:
  - The PIN_FLD_POID field contains the POID of the **/admin_action/suspended_usage/edit** object.
  - The PIN_FLD_COUNT field contains the number of records that were processed.

- **1** if the **/admin_action/suspended_usage/edit** POID in the input flist does not match the POID at the top of the **/suspended_usage_edits** object stack.

Deleting Records for Suspended EDRs

Use PCM_OP_SUSPENSE_DELETE_USAGE to delete records for suspended EDRs.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

The Suspense Management Center calls this opcode to delete EDRs. EDRs can be deleted only if their status is **Written-off** or **Succeeded**.

PCM_OP_SUSPENSE_DELETE_USAGE takes as input an array of **/suspended_usage** object POIDs to be deleted.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_DELETE_USAGE performs these operations:

- Confirms that the suspended EDR has a status of **Succeeded** or **Written off.** The value of the PIN_FLD_STATUS field in the **/suspended_usage** object must be **2** or **3**.
- Deletes the **/suspended_usage** object.
- Generates a **/event/notification/suspense/delete** object that records the deletion.

PCM_OP_SUSPENSE_DELETE_USAGE returns the routing POID specified in the input flist.

Deleting Records for Suspended Batches

Use PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES to delete **/suspended_batch** objects (Batch Suspense Records).

> **Note:**
>
> This opcode deletes records, not the files associated with them.

Suspense Management Center calls this opcode to delete Batch Suspense Records (**/suspended_batch** objects). Batch Suspense Records can be deleted only if their status is **Written-off** or **Succeeded**.

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES takes as input an array of **/suspended_batch** object POIDs to be deleted.

For each **/suspended_batch** object specified in the input flist, PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES performs these operations:

- Creates a transaction if it is not already opened.
- Confirms that the suspended batch file has a status of **Succeeded** or **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_batch** object must be **2** or **3**. Otherwise, an error code is generated and the transaction ends.

- Deletes the **/suspended_batch** object.
- Generates an **/event/notification/suspense/batch_delete** event.
- Closes the transaction.

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES returns the routing POID specified in the input flist.

Deleting Call Records with a Specific Recycle Key and a Status of Succeeded or Written-Off

Use PCM_OP_SUSPENSE_SEARCH_DELETE to delete call records with a specific recycle key and a status of **Succeeded** or **Written-off**.

Set the **PIN_FLD_FLAGS** field to either of these values:

- **0**: Directs this opcode to delete EDRs with a status of:
  - **Succeeded** (successfully processed)
  - Written-off
- **1**: Directs this opcode to delete EDRs with a status of:
  - **Succeeded** (successfully processed)
  - Written-off
  - **Suspended**. This opcode first writes off, then deletes, these EDRs

Deleting EDRs in a CDR File

PCM_OP_SUSPENSE_SEARCH_DELETE is used with the BRM standard recycling feature, which acts on all the calls contained in a single CDR file simultaneously. Using **pin_recycle** with the **-d** parameter deletes all calls in a CDR file that have a status of **Succeeded**. Using **pin_recycle** with the **-D** parameter deletes all calls in a CDR file with a status of **Succeeded** or **Written off**.

Deleting Calls with a Recycle Key

PCM_OP_SUSPENSE_SEARCH_DELETE is also used by features that need to temporarily delay rating of EDRs. These features include pipeline modules to add a recycle key and error to those EDRs during pipeline rating. The error prevents EDRs from being rated by assigning them a status of **Suspended**. Those features then use PCM_OP_SUSPENSE_EDIT_USAGE to find and recycle the call records. If successful, the EDRs are assigned a status of **Succeeded**. This opcode then deletes those successfully recycled call records.

Writing Off Suspended EDRs

Use PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE to write off suspended EDRs.

When a suspended EDR is written off, it can no longer be edited or recycled.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE takes as input an array of **/suspended_usage** object POIDs.

- PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE creates an **/admin_action/suspended_usage/writeoff** object that records the write-off.

For each **/suspended_usage** object specified in the input flist, PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE performs these operations:

- Confirms that the suspended EDR has a status of **Suspended**; the PIN_FLD_STATUS value must be **0**.
- Adds the POID of the newly created **/admin_action/suspended_usage/writeoff** object to the array of actions in the **/suspended_usage** object.
- Changes the status to **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is changed to **3**.
- Generates an **/event/notification/suspense/batch_writeoff** event.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE returns the POID of the **/admin_action/suspended_usage/writeoff** object created.

Writing Off Suspended Batches

Use PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES to write off suspended CDR files.

When you write off a suspended CDR file, you can no longer resubmit it, but you can delete it.

The opcode creates an **/admin_action/suspended_batch/writeoff** object that records the write-off and sets the status of the Batch Suspense Record (**/suspended_batch**) to **Written-off**.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES performs these operations:

- Create a transaction if it is not already opened.
- Confirms that the suspended EDR has a status of **Suspended**; the PIN_FLD_STATUS value must be **0**.
- PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES takes as input an array of **/suspended_batch** object POIDs.
- Adds the POID of the newly created **/admin_action/suspended_batch/writeoff** object to the array of actions in the **/suspended_batch** object.
- Changes the status of the Batch Suspense Record (**/suspended_batch**) to **Written off**; The value of the PIN_FLD_STATUS field in **/suspended_batch** is changed to **3**.
- Generates an **/event/notification/suspense/batch_writeoff** event.
- Closes the transaction.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES returns the POID of the **/admin_action/suspended_batch/writeoff** object created.

Processing Suspended Records in Bulk

You can use the Suspense Manager opcodes to edit, delete, recycle, and write off a large number of suspended records. For more information, see:

- Processing Suspended Records in Multiple Steps
- Editing Suspended Records in Bulk
- Writing Off Suspended Records in Bulk
- Deleting Suspended Records in Bulk

Processing Suspended Records in Multiple Steps

You can process the suspended records in multiple steps by calling the opcodes multiple times, to avoid a large database transaction:

1. Specify the number of records to process in each opcode call in a configuration file and load the file into the **/config/pin_suspense_system_params** storable class.

   For more information, see "Configuring the Number of Suspended Records to Process in a Transaction".
2. Call the opcode in calc-only mode to retrieve the count and POID range of the records that match your search criteria.
3. Use a simple logic to determine the number of times to call the opcodes depending on the number of records you want each opcode call to process.
4. Call the opcode several times with a set of records each time and consolidate the results returned by each opcode call.

> **Note:**
>
> For each opcode call, you must provide the POID range and the corresponding arguments in the input flist.

Because each operation is performed in multiple steps, if the operation is successful in any of the steps, you get the number of records processed. You also get an error message for the unsuccessful records, which you can display to the user. If any one of the steps fails, the entire step and the following steps are canceled and an error message is returned.

> **Note:**
>
> Suspense Management Center and the **pin_recycle** utility perform suspense management operations in multiple steps by calling the opcodes multiple times.

Editing Suspended Records in Bulk

Use the PCM_OP_SUSPENSE_SEARCH_EDIT opcode to perform the same set of edits on a large number of suspended records that meet the criteria you specify.

This opcode makes changes to the records in one database operation instead of accessing the database for each record. It calls the following opcodes:

- PCM_OP_BULK_WRITE_FLDS, to update the objects in the database.
- PCM_OP_ACT_USAGE, to generate the edit event notification.

> **Note:**
>
> You cannot undo edits performed on a large number of records or any edits made before the bulk edit operation.

PCM_OP_SUSPENSE_SEARCH_EDIT follows these steps to edit suspended records:

1. Takes as input the following information:
   - The POID type of the suspended usage class.
   - The search criteria template.
   - The fields and values that need to be edited in the object.
   - An array of the EDR fields to be edited, including the old and new values.
2. Does one of the following:
   a. If the PCM_OPFLG_CALC_ONLY flag is set, returns the count and the POID range of records that meet the search criteria.
   b. If the PCM_OPFLG_CALC_ONLY flag is not set, searches for the **/suspended_usage_edits** objects that correspond to the account, which is determined by the login session.
3. Does one of the following:
   a. If it finds **/suspended_usage_edits** objects for the specified account, clears all the POIDs of the edit actions stored in the objects.

   > **Note:**
   >
   > After the objects are changed, the current changes or previous changes cannot be undone.

   b. If it does not find a **/suspended_usage_edits** object for the appropriate account, creates a new **/suspended_usage_edits** object.
4. For each EDR field to be edited, creates an **/admin_action/suspended_usage/edit** object, which includes both the old and new values.
5. For each **/suspended_usage** object, performs the following operations:
   - Verifies that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
   - Adds the **/admin_action/suspended_usage/edit** object POID and the old value to each **/suspended_usage** object's action array.
   - Updates the PIN_FLD_EDITED field to indicate that the field has been edited.

If successful, PCM_OP_SUSPENSE_SEARCH_EDIT generates an edit notification event that includes the administrative action POID of the edit action and returns success along with the

count of the objects edited. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Writing Off Suspended Records in Bulk

Use the PCM_OP_SUSPENSE_SEARCH_WRITE_OFF opcode to write off all suspended records that meet the criteria you define.

> **Note:**
>
> You cannot edit or recycle suspended records that are written off.

This opcode writes off a large set of suspended records in one database operation instead of accessing the database for each record. It calls the following opcodes:

- PCM_OP_BULK_WRITE_FLDS to mark a large number of objects in the database as written off.
- PCM_OP_ACT_USAGE to generate the write-off event notification.

PCM_OP_SUSPENSE_SEARCH_WRITE_OFF follows these steps to write off suspended records:

1. Takes as input the POID type of the suspended usage class and the search criteria template for the objects to be written off.
2. If the PCM_OPFLG_CALC_ONLY opcode flag is set, returns the count of records that match the search criteria and the POID range of the records that satisfy the criteria.
3. If the PCM_OPFLG_CALC_ONLY flag is *not* set, creates the **/event/notification/suspense/batch_writeoff** object that records the write off and returns that object with the count of records written off.
4. For each **/suspended_usage** object that meets the criteria specified in the template, performs these operations:
   - Verifies that the suspended EDR has a status of **Suspended**. The PIN_FLD_STATUS value in the object must be **0**.
   - Adds the POID of the newly created **/admin_action/suspended_usage/writeoff** object to the array of actions in the **/suspended_usage** object.
   - Changes the status to **Written-off**. The value of the PIN_FLD_STATUS field in the **/suspended_usage** object is changed to **3**.

If successful, PCM_OP_SUSPENSE_SEARCH_WRITE_OFF generates a write-off notification event that includes the administrative action POID of the write-off action and returns success along with the number of records written off. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

Deleting Suspended Records in Bulk

Use the PCM_OP_SUSPENSE_SEARCH_DELETE opcode to delete all suspended records that meet the criteria you define.

> **Note:**
>
> You can only delete records that are succeeded or written off.

This opcode deletes a large set of suspended records in one database operation instead of accessing the database for each record. It calls PCM_OP_ACT_USAGE to generate the delete event notification.

PCM_OP_SUSPENSE_SEARCH_DELETE follows these steps to delete suspended records:

1. Takes as input the POID type of the suspended usage class and the search criteria template for the objects to be deleted.
2. If the PCM_OPFLG_CALC_ONLY opcode flag is set, returns the count of records that match the search criteria and the POID range of the records that satisfy the criteria.

   If the PCM_OPFLG_CALC_ONLY flag is *not* set, creates the **/event/notification/suspense/batch_delete** object that records the deletion.
3. For each **/suspended_usage** object that meets the criteria specified in the template, performs these operations:
   - Verifies that the suspended EDR has a status of **Succeeded** or **Written off**. The PIN_FLD_STATUS value in the object must be **2** or **3**.
   - EDRs with a status of **Succeeded** and those with a status of **Written off** that do not have a deferred duration are deleted immediately.
   - If the status of the suspended record is **Written off**, the opcode checks if there is a deferred duration and if the deferred duration is greater than 0. The deferred duration is a parameter defined in the **/config/suspense_params** object.

     > **Note:**
     >
     > The load utility is provided to load the deferred duration parameter.

     If there is a deferred duration, then the PCM_OP_SUSPENSE_SEARCH_DELETE opcode will not remove the suspended record from the database but rather change the status of the suspended records from written off to delete pending. The PCM_OP_SUSPENSE_SEARCH_DELETE opcode will create a schedule object to run the PCM_OP_SUSPENSE_DEFERRED_DELETE opcode.
   - The schedule object will call the PCM_OP_SUSPENSE_DEFERRED_DELETE opcode to be run at the deferred duration time.
   - The PCM_OP_SUSPENSE_SEARCH_DELETE or PCM_OP_SUSPENSE_DEFERRED_DELETE opcode deletes the object from the suspense db.
4. Generates a **/event/notification/suspense/delete** object that records the deletion for each suspended record that was deleted.

If successful, PCM_OP_SUSPENSE_SEARCH_DELETE generates a delete notification event that includes the administrative action POID of the delete action and returns success along with the number of records deleted. If the operation fails in any record, it cancels the entire operation and returns failure with the appropriate error code, leaving the state of the record as it was before the operation.

52
Suspense Management Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Suspense Management utilities.

load_edr_field_mapping

Use this utility to load EDR field mapping into the **/edr_field_mapping** object in the BRM database.

Location

*BRM_home***/bin**

Syntax

```
load_edr_field_mapping [-d] [-v] [-t] [-h] XML_file
```

Parameters

-d

Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v

Displays detailed information as the utility runs.

-t

Validates the XML file. Your EDR field mapping XML file must conform to the XML schema rules in the *BRM_home***/xsd/edr_field_mapping.xsd** file.

-h

Shows the help on the utility.

XML_file

The name and location of the EDR field mapping configuration file, which maps the EDR field name to an ID number. The default **edr_field_mapping.xml** file is in *BRM_home***/sys/data/config** directory.

If you copy the **edr_field_mapping.xml** file to the same directory from which you run the **load_edr_field_mapping** utility, you do not have to specify either the path or the file name.

If you run the command in a different directory from where the **edr_field_mapping.xml** file is located, you must include the entire path for the file.

load_pin_suspense_editable_flds

Use this utility to load editable fields into the **/config/suspense_editable_flds** object in the BRM database. You define editable fields in the **pin_suspense_editable_flds** file in *BRM_home***/sys/data/config**.

> **Caution:**
>
> The **load_pin_suspense_editable_flds** utility overwrites existing **/config/suspense_editable_flds** objects. If you are updating editable fields, you cannot load new editable fields only. You must load complete sets of editable fields each time you run the utility.

> **Note:**
>
> To connect to the BRM database, the **load_pin_suspense_editable_flds** utility needs a configuration file in the directory from which you run the utility.

Location

*BRM_home***/bin**

Syntax

```
load_pin_suspense_editable_flds [-d] [-v] [pin_suspense_editable_flds_file]
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_editable_flds_file
The name and location of the file that defines the list of editable fields used by Suspense Management Center. The default **pin_suspense_editable_flds** file is in *BRM_home***/sys/data/config**.
If you copy the **pin_suspense_editable_flds** file to the same directory from which you run the **load_pin_suspense_editable_flds** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_suspense_editable_flds** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_editable_flds** utility notifies you when it successfully creates the **/config/suspense_editable_flds** object. Otherwise, look in the **default.pinlog** file for errors.

This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_editable_flds** object by using Object Browser or the **robj** command with the **testnap** utility.

> **Note:**
>
> You must restart Suspense Management Center to make new editable fields available.

load_pin_suspense_override_reason

Use this utility to load Pipeline Manager override reasons into the **/config/suspense_override_codes** object in the BRM database. You define Pipeline Manager override reasons in the **pin_suspense_override_reason** file in *BRM_home***/sys/data/config**.

For more information, see "Overriding Pipeline Suspense Handling Rules".

> **Caution:**
>
> The **load_pin_suspense_override_reason** utility overwrites existing suspense override reasons. You must load complete sets of override reasons each time you run the utility.

> **Note:**
>
> To connect to the BRM database, the **load_pin_suspense_override_reason** utility needs a configuration file in the directory from which you run the utility.

Location

*BRM_home***/bin**

Syntax

```
load_pin_suspense_override_reason [-d] [-v] pin_suspense_override_reason_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_override_reason_file
The name and location of the file that defines the override reasons. The default **pin_suspense_override_reason** file is in *BRM_home***/sys/data/config**.
If you copy the **pin_suspense_override_reason** file to the same directory from which you run the **load_pin_suspense_override_reason** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_suspense_override_reason** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_override_reason** utility notifies you when it successfully creates the **/config/suspense_override_codes** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_override_codes** object by using Object Browser or the **robj** command with the **testnap** utility.

> **Note:**
>
> You must restart Suspense Management Center to enable it to use the new suspense override reasons.

load_pin_suspense_params

Use this utility to load system-level configuration information for Suspense Manager into the **/config/suspense_params** object in the BRM database. You define the system parameters for Suspense Manager, such as the number of records to process in each opcode call, in the **pin_suspense_params** file in the *BRM_home***/sys/data/config** directory.

For more information, see "Configuring the Number of Suspended Records to Process in a Transaction" and "Processing Suspended Records in Multiple Steps".

> **Caution:**
>
> The **load_pin_suspense_params** utility overwrites existing Suspense Manager system parameters. You must load complete sets of parameters each time you run the utility.

> **Note:**
>
> To connect to the BRM database, the **load_pin_suspense_params** utility needs a configuration file in the directory from which you run the utility.

Location

BRM_home/bin

Syntax

```
load_pin_suspense_params [-d] [-v] filename
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

filename
The name of the text file containing the configuration parameters for suspense management. The default file name is **pin_suspense_params**.

Results

The **load_pin_suspense_params** utility notifies you when it successfully creates the **/config/suspense_params** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the data was loaded, display the **/config/suspense_params** object by using Object Browser or the **robj** command with the **testnap** utility.

load_pin_suspense_reason_code

Use this utility to load suspense reasons and subreasons into the /**config/suspense_reason_code** object in the BRM database. You define suspense reasons and subreasons in the **pin_suspense_reason_code** file in *BRM_home*/**sys/data/config/suspense_reason_code**.

For more information, see "Changing the List of Suspense Reasons and Subreasons".

> **Caution:**
>
> The **load_pin_suspense_reason_code** utility overwrites existing suspense reason and subreason codes. If you are updating suspense reason and subreason codes, you cannot load new codes only. You must load complete sets of codes each time you run the utility.

> **Note:**
>
> To connect to the BRM database, the **load_pin_suspense_reason_code** utility needs a configuration file in the directory from which you run the utility.

Location

BRM_home/**bin**

Syntax

```
load_pin_suspense_reason_code [-d] [-v] pin_suspense_reason_code_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

pin_suspense_reason_code_file
The name and location of the file that defines suspense reasons and subreasons. The default **pin_suspense_reason_code** file is in *BRM_home*/**sys/data/config**.
If you copy the **pin_suspense_reason_code** file to the same directory from which you run the **load_pin_suspense_reason_code** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_suspense_reason_code** file is located, you must include the entire path for the file.

Results

The **load_pin_suspense_reason_code** utility notifies you when it successfully creates the **/config/suspense_reason_code** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the network elements were loaded, display the **/config/suspense_reason_code** object by using Object Browser or the **robj** command with the **testnap** utility.

> **Note:**
>
> You must restart Pipeline Manager to make the new suspense reason codes available.

> **Note:**
>
> If you are changing the suspense reason or subreason codes, you must also modify the **suspense_reason_code.en_US** file and run the **load_localized_strings** utility. See "Installing and Configuring Suspense Manager".

load_pin_batch_suspense_override_reason

Use the **load_pin_batch_suspense_override_reason** utility to load batch suspense override-able reason codes into the **/config/batch_suspense_override_reason** object in the BRM database. You define batch suspense override reason codes in the **pin_batch_suspense_override_reason** file in *BRM_home***/sys/data/config**. By default, no reason can be overridden, so the file is a placeholder.

> **Caution:**
>
> The **load_pin_batch_suspense_override_reason** utility overwrites the existing **/config/batch_suspense_override_reason** object in the BRM database. If you are updating the **/config/batch_suspense_override_reason** object, you must load complete sets of batch suspense override-able reasons each time.

> **Note:**
>
> To connect to the BRM database, the utility needs the Connection Manager (CM) to be up and running.

Location

*BRM_home***/bin**

Syntax

```
load_pin_batch_suspense_override_reason [-d] [-v]
pin_batch_suspense_override_reason_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_batch_suspense_override_reason** *any_other_parameter* **-v >** *filename***.log**

pin_batch_suspense_override_reason_file
The name and location of the file that defines batch suspense override-able reason codes. The default **pin_batch_suspense_override_reason** file is in *BRM_home***/sys/data/config**. If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

> **Tip:**
>
> If you copy the **pin_batch_suspense_override_reason** file to the directory from which you run the **load_pin_batch_suspense_override_reason** utility, you don't have to specify the path or file name. The file must be named **pin_batch_suspense_override_reason**.

Results

If the utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the override reason codes were loaded, display the **/config/batch_suspense_override_reason** object by using Object Browser or the **robj** command with the **testnap** utility.

The following is an example of a **pin_batch_suspense_override_reason** file, which would be an input for this utility and could be compared to the output:

```
# Override Suspense Reason000010000200003
```

load_pin_batch_suspense_reason_code

Use the **load_pin_batch_suspense_reason_code** utility to load batch suspense reason codes into the **/config/batch_suspense_reason_code** object in the BRM database. You define batch suspense reasons in the **pin_batch_suspense_reason_code** file in *BRM_home***/sys/data/config**. BRM uses suspense reason codes to load suspense reasons into a batch suspense record when a call details record (CDR) file is suspended.

> **Caution:**
>
> The **load_pin_batch_suspense_reason_code** utility overwrites the existing **/config/batch_suspense_reason_code** object in the BRM database. If you are updating load batch suspense reason codes, you cannot load new batch suspense reason codes only. Therefore, you must load a complete set of load batch suspense reason codes each time you run the utility.

> **Note:**
>
> The **load_pin_batch_suspense_reason_code** utility must be connected to a running CM to load batch suspense reason codes into the Infranet database.

Location

*BRM_home***/bin**

Syntax

```
load_pin_batch_suspense_reason_code [-d] [-v]
pin_batch_suspense_reason_code_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_batch_suspense_reason_code** *any_other_parameter* **-v >** *filename***.log**

pin_batch_suspense_reason_code_file
The name and location of the file that defines the batch suspense reason codes. The default **pin_batch_suspense_reason_code** file is in *BRM_home***/sys/data/config**.
If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

> **Tip:**
>
> If you copy the **pin_batch_suspense_reason_code** file to the directory from which you run the **load_pin_batch_suspense_reason_code** utility, you do not have to specify the path or file name. The file must be named **pin_batch_suspense_reason_code**.

Results

The **load_pin_batch_suspense_reason_code** utility notifies you when it successfully creates the **/config/batch_suspense_reason_code** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that the elements were loaded, display the **/config/batch_suspense_reason_code** object by using Object Browser or the **robj** command with the **testnap** utility.

The following example shows sample entries from the **/config/batch_suspense_reason_code** object:

```
PIN_FLD_POID                      POID [0] 0.0.0.1 /config/batch_suspense_reason_code 93712 0
0 PIN_FLD_CREATED_T               TSTAMP [0] (1153474255) 21/07/2006 15:00:55:000 PM
0 PIN_FLD_MOD_T                   TSTAMP [0] (1153474255) 21/07/2006 15:00:55:000 PM
0 PIN_FLD_READ_ACCESS                STR [0] "G"
0 PIN_FLD_WRITE_ACCESS               STR [0] "S"
0 PIN_FLD_ACCOUNT_OBJ               POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_DESCR                      STR [0] ""
0 PIN_FLD_HOSTNAME                   STR [0] "-"
0 PIN_FLD_NAME                       STR [0] "batch_suspense_reason_code"
0 PIN_FLD_OP_CORRELATION_ID          STR [0] "2:CT1255:UnknownProgramName:0:AWT-
EventQueue-0:3:1153836497:0:root.0.0.0.1::user1:123456789"
0 PIN_FLD_PROGRAM_NAME               STR [0] "load_pin_batch_suspense_reason_code"
0 PIN_FLD_VALUE                      STR [0] ""
0 PIN_FLD_VERSION                    STR [0] "1"
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [0] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 0
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [471] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 1
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [119] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [120] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [126] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [127] allocated 1, used 1|
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [147] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
0 PIN_FLD_SUSPENSE_REASONS         ARRAY [148] allocated 1, used 1
1     PIN_FLD_SUSPENSE_REASON       ENUM [0] 2
```

53
Recycling EDRs in Pipeline-Only Systems

This chapter describes how to configure and use EDR recycling. EDR recycling is the Oracle Communications Billing and Revenue Management (BRM) feature used by systems that use Pipeline Manager, but do not store suspended EDRs in the BRM database. The pipeline-only recycling feature uses the FCT_PreRecycle module to mark EDRs for recycling, and the FCT_Recycle module to send the rejected EDRs to a file for processing by hand.

Systems using BRM with Pipeline Manager use either the standard recycling tools or the Suspense Manager service integration component for recycling and deleting EDRs.

For details, see "About the EDR Recycling Features".

For details on how to use FCT_Reject to reject EDRs, see "FCT_Reject".

Before reading this document, you should be familiar with how Pipeline Manager works and how to configure it.

About Recycling EDRs

When processing a CDR file, there might be non-valid EDRs in the file, or your pipelines might not be set up correctly to handle certain EDRs. You use EDR recycling to fix configuration problems and re-process EDRs.

The recycling process uses these pipeline modules:

- FCT_Reject
- FCT_PreRecycle
- FCT_Recycle

Overview of EDR recycling:

1. You start Pipeline Manager with the FCT_PreRecycle, FCT_Recycle, and FCT_Reject modules active. (The FCT_PreRecycle and FCT_Recycle modules do nothing until you start the recycle process by using a semaphore.)
2. When an EDR is processed, a module may find an error in the EDR. The error is appended to the EDR, and a flag is set to indicate that the EDR has an error. The EDR is sent to the next module. Each module adds errors, if any more are found.
3. The FCT_Reject module analyzes the errors in the EDR. If necessary, the EDR is moved to a reject file.
4. You examine the errors and determine how to reconfigure Pipeline Manager to prevent the errors.
5. You use a semaphore file entry to start the pre-recycling process. This sends the rejected EDRs through the pipeline again. The FCT_PreRecycle module adds a flag to the EDR to let the other modules know that the EDR is being recycled.

   You can pre-recycle and recycle EDRs in test mode or real mode. Typically, you run the pre-recycle and recycling processes in test mode first, to see if the errors have been fixed. When there are no longer any errors, you pre-recycle and recycle in real mode.
6. The FCT_Recycle module runs at the end of the pipeline It does one of the following:

- In test mode, the module creates a report about the processing, but does not send the EDRs to an output file.
- In recycle mode, the module sends the results to an output file, and attaches a sequence number to the output file.

> **Note:**
>
> You can configure the output module to send an entire file to the error directory if it includes a lot of errors. You can configure the threshold for the number of errors allowed per file. See "Specifying the Maximum Errors Allowed in an Input File".

How the FCT_Reject Module Works

The FCT_Reject module must be run after all rating and enrichment modules. It should be run as the second-to-last function module in the pipeline (the last function module is the FCT_Suspense module). This is because all potential errors must be found before the FCT_Reject module processes the EDRs.

You can run the FCT_Reject module from the registry or by using a semaphore file entry.

The FCT_Reject module does the following:

1. The FCT_Reject module checks the error status of the EDR. If the EDR contains an error status with a *warning* or *critical* severity, the EDR is rejected. The FCT_Reject module changes the value of the DETAIL.DISCARDING field from 0 to 1.

   > **Note:**
   >
   > If the DETAIL.DISCARDING field is already set to 1, the EDR was rejected in a previous pass through the pipeline, and is rejected again.

   If the error type in the EDR is not identified in the registry **StreamMap** entry, the EDR is sent to default reject stream.

2. By default, the EDR is moved to the reject stream, as identified in the **RejectStream** registry entry. The EDR is stored in a file that is used by the recycling modules. EDRs can also be rejected in the recycle process.

   If the reject stream is not specified, the EDR is moved to the normal output stream, but the discard field is set to **1**, indicating that the EDR has been rejected.

Using a Reject Output Stream

Use the **UseRejectStream** entry to specify how to handle rejected EDRs. You can do the following:

> **Note:**
>
> If you use **UseRejectStream**, you must use the **StreamMap** entry.

Specifies whether to use the reject output stream:

- **True**. Rejected EDRs are sent to the reject stream.
- **False**. Rejected EDRs are sent to the normal output stream, but flagged as discarded.

Specifying Multiple Reject Streams

By default, rejected EDRs are sent to a single reject stream. However, you can use the **StreamMap** registry entry to specify separate reject streams for different types of errors.

> **Note:**
>
> If you use **StreamMap**, you must use the **UseRejectStream** entry.

For example, this entry sends errored TAP records to the output stream named **Rap0101Output**.

```
ERR_TAP3_RET = Rap0101Output
```

The output stream must be configured.

Recycling Assembled EDRs

If you use both the FCT_CallAssembling module and the FCT_Reject module in a pipeline, use the optional FCT_Reject module **CallAssemblingModule** registry entry to ensure that the complete EDRs are recycled. Otherwise, only part of the EDR is recycled.

The FCT_Reject **CallAssemblingModule** registry entry is a pointer to the FCT_CallAssembling module, for example:

```
CallAssemblingModule = ifw.Pipelines.Pipe.Functions.Standard.FunctionPool.CallAssembling
```

Processing EDRs with Errors

Use the FCT_Reject **MinErrorSeverity** registry entry to reject EDRs that have a specified severity. This allows the EDR to be processed with warning or normal error messages without being rejected.

You can specify one of the following:

- **-1** = undef
- **0** = debug
- **1** = normal
- **2** = warning

- **3** = minor
- **4** = major
- **5** = critical

To allow warning and normal messages without rejecting the EDR, set this entry to **3**. Valid values for **MinErrorSeverity** are 3, 4, and 5.

By default, this entry is not used.

How the FCT_PreRecycle Module Works

The FCT_PreRecycle module is always the first module in the pipeline.

Although you can *activate* the FCT_PreRecycle module from the startup registry, you cannot run the FCT_PreRecycle module from the startup registry; you must *run* it by using a semaphore file.

The FCT_PreRecycle module does the following:

1. The module gets the file of rejected EDRs from the reject stream output directory.
2. The module puts the reject EDR file into the input directory for recycling. It uses the same input directory as the incoming CDR files. It adds a recycle suffix to the file and a sequence number, so the original input file in the output directory cannot be overwritten.

   You can recycle all EDRs in the reject directory, or list specific files to recycle. See "Recycling EDRs".
3. For each EDR to recycle, the module sets a value in the INTERN_PROCESS_STATUS field to indicate that the EDR is being recycled. This tells the FCT_Recycle module which EDRs to process, and allows the discounting modules to recalculate discount amounts correctly.
   - The value is set to **1** if the EDR is being recycled.
   - The value is set to **2** if the recycling is in test mode.

You can recycle all EDRs in the reject directory, or list specific files to recycle. See "Sample Semaphore File Entries".

How the FCT_Recycle Module Works

The FCT_Recycle module is the last function module in the pipeline, before the output.

You activate the FCT_Recycle module from the startup registry, but it does nothing until the FCT_PreRecycle module starts the recycling process.

The FCT_Recycle module reads the INTERN_PROCESS_STATUS field for each EDR.

- If the value is **2**, recycling is in test mode. The FCT_Recycle module doesn't send the EDRs to an output directory. Instead, the FCT_Recycle module creates a report with the following data:
  - Stream name.
  - Total number of processed EDRs.
  - Number of EDRs that can be recycled without an error.
  - Number of EDRs that still generate an error.
  - List of all errors.

  - The wholesale charge amount from all successfully recycled EDRs. (This data is taken from the WHOLESALE_CHARGE field.)
  - The wholesale charge amount from all EDRs that still have errors. (This data is taken from the WHOLESALE_CHARGE field.)
  - The total duration for all successfully recycled EDRs. (This data is taken from the DURATION_MINUTES field.)
  - The total duration from all EDRs that still have errors. (This data is taken from the DURATION_MINUTES field.)

  You can use this data to determine if the EDRs are worth further configuration and processing.

- If the value is **1**, recycling occurs. All EDRs are processed as usual, with the following differences in comparison to normal input file processing:
  - A sequence number is generated.
  - The sequence offset value is generated.
  - The sequence check is inactivated.

Testing Recycling EDRs

Once you have determined that EDRs have been rejected, the first step is to correct any pipeline problems that caused the problem. After that you usually test recycle the CDR file to ensure that the changes have had the desired affect.

Follow these steps to test recycle EDRs:

1. Configure the FCT_Reject module. See "FCT_Reject".

   Typically, rejected EDRs are sent to the reject stream. You configure the reject stream in the registry in the following places:

   - In the FCT_Reject module pipeline configuration
   - In the Output stream

   For a sample Output stream configuration see "Sample Output Configuration".

   > **Note:**
   >
   > When you test recycling, first inactivate the FCT_Reject module.

2. Configure the FCT_PreRecycle module. See "FCT_PreRecycle".

   You configure the reject stream in the registry in the following places:

   - In the FCT_PreRecycle module pipeline configuration.
   - In the input stream.

   The module uses the same input configuration as the incoming CDR files, so you don't need to configure a separate input stream.

3. Configure the FCT_Recycle module. See "FCT_Recycle".

   Configure the FCT_Recycle module RecycleLog registry entry to specify the message file parameters. These settings are specified in the **ProcessLog** registry section. For more information, see "LOG".

4. Use a semaphore to inactivate the FCT_Reject module:

```
Module
Reject.Module.Active = False
```

5. Use a semaphore to run the FCT_PreRecycle module in test mode:

```
Recycle.Module.RecycleTest {}
```

6. Review the log files that you configured in FCT_Recycle for errors, and repeat these steps as necessary.

Recycling EDRs

When you finish test recycling EDRs, follow these steps to do the actual recycling:

1. Configure the FCT_Reject module. See "FCT_Reject".

   Typically, rejected EDRs are sent to the reject stream. You configure the reject stream in the registry in the following places:

   - In the FCT_Reject module pipeline configuration
   - In the Output stream

   For a sample Output stream configuration see "Sample Output Configuration".

   > **Note:**
   >
   > When you test recycling, first inactivate the FCT_Reject module.

2. Configure the FCT_PreRecycle module. See "FCT_PreRecycle".

   You configure the reject stream in the registry in the following places:

   - In the FCT_PreRecycle module pipeline configuration.
   - In the input stream.

   The module uses the same input configuration as the incoming CDR files, so you don't need to configure a separate input stream.

3. Configure the FCT_Recycle module. See "FCT_Recycle".

   Configure the FCT_Recycle module **RecycleLog** registry entry to specify the message file parameters. These settings are specified in the **ProcessLog** registry section. For more information, see "LOG".

4. Use a semaphore to run the FCT_PreRecycle module.

   You can recycle all EDRs in the reject directory, or list specific files to recycle. See "Sample Semaphore File Entries".

5. Review the log files that you configured in FCT_Reject for errors.

Part VI
Loading Rated Events

This part describes how to load rated events when using Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. It contains the following chapters:

- Understanding Rated Event Loader
- Installing Rated Event Loader
- Configuring Rated Event Loader
- Loading Prerated Events

54

Understanding Rated Event Loader

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader and how it imports pipeline-rated events into the BRM database.

About RE Loader

RE Loader is an optional BRM application that loads pipeline-rated events directly into the BRM database, bypassing the Connection Manager (CM) and Data Manager (DM). RE Loader then updates account balances, billing items, and journals in the BRM database. After the events are loaded, you can run BRM applications such as billing and reports on the rated data.

About the Database Schema

RE Loader uses a partitioned database and inserts prerated events into separate partitions.

> **Note:**
>
> You *must* partition your database when loading prerated events.

About RE Loader Event Types

By default, RE Loader loads the wireless services and corresponding BRM event types shown in Table 54-1.

However, you can configure RE Loader to load custom events. For information, see "Configuring the RE Loader Infranet.properties File".

Table 54-1 Service and Event Types Loaded by Default

Service Type	Event Type
GPRS (General Packet Radio Service)	**/event/delayed/session/gprs**
GSM (Global System for Mobile Communication)	**/event/delayed/session/telco/gsm**

These events are loaded into separate BRM database partitions allocated for delayed events. The event types are called "delayed" because they are rated before they are loaded, and there is a delay between the two actions. This is unlike events loaded in real time or by Universal Event (UE) Loader.

> **Note:**
>
> You should not load the same event types by using RE Loader and another method such as an optional manager or UE Loader.

About Loading Prerated Events

Prerated events are events that have been rated by Pipeline Manager prior to being loaded into the BRM database. Basic steps of pipeline rating and event loading include:

1. Pipeline Manager rates events associated with call detail records (CDRs).

   For information on how Pipeline Manager prerates events, see "How Events Are Rated by Using Pipeline Manager".

2. Pipeline Manager creates an output file for each service type and places them in one or more output directories.

   You configure the number and location of your pipeline output directories by using the pipeline EXT_OutFileManager module.

3. RE Loader loads the prerated events.

   For information, see "RE Loader Process Overview".

About Loading Rerated Events

It is possible to discover pricing or rating configuration errors after events have been rated by Pipeline Manager and loaded into the BRM database. When this occurs, you rerate any incorrectly rated events and reload them into the BRM database.

When you need to rerate and reload pipeline-rated events, you must:

1. Extract events that need rerating from the BRM database by using the Event Extraction Tool.

   > **Note:**
   >
   > Event Extraction Manager is included in the RE Loader installation.

2. Rerate those events by using Pipeline Manager. Pipeline Manager backs out the previous rating changes and then rerates the events.
3. Reload the rerated events by using RE Loader.

   For information, see "RE Loader Process Overview".

RE Loader Process Overview

RE Loader processes output files generated from Pipeline Manager. You send these files to RE Loader manually through a command-line utility or automatically by the Batch Controller.

After RE Loader receives a pipeline output file, it:

1. Checks the event header to determine the storable class type and whether the file contains prerated, rerated, or discount events.
2. Parses the event data record (EDR) data into multiple temporary files, one for each BRM database table to be loaded, and places the files in a temporary directory.
3. Loads events from each temporary file into the BRM database by using multiple Oracle SQL Loader utility **sqlldr** processes.

4. Calls stored procedures to update the account balances, bill items, and journals.
5. Logs the session information in its log file (**rel.pinlog**).

About Running RE Loader

You can run RE Loader in one of the following ways:

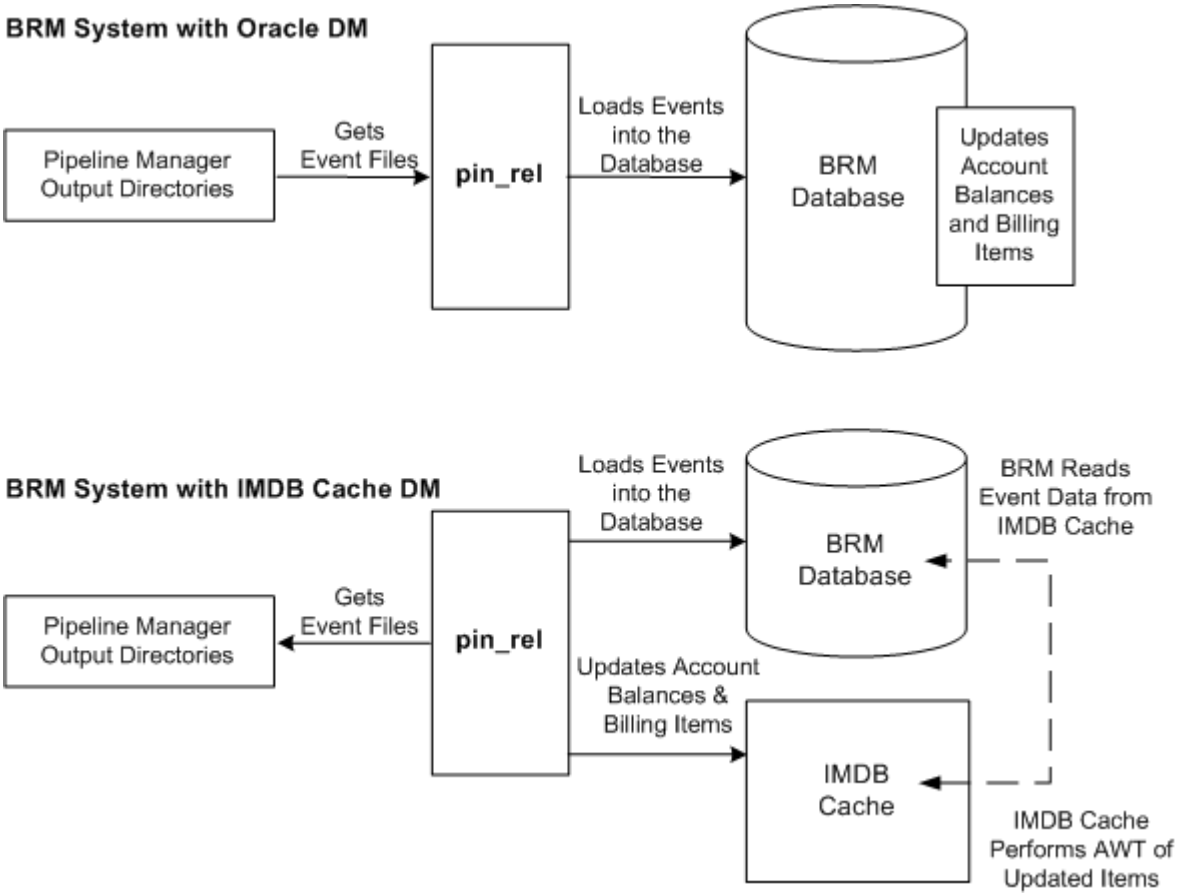
- Manually from a command line. See "About Running RE Loader Manually".
- Automatically by using Batch Controller and the RE Loader batch handler. See "About Running RE Loader Automatically".
- As a daemon. See "About Running the RE Loader Daemon".

About Running RE Loader Manually

When you run RE Loader manually from a command line, you specify the location of the pipeline output file in the command line.

Figure 54-1 shows the RE Loader work flow when you run it manually from a command line:

Figure 54-1 Work Flow of Manual Execution of RE Loader



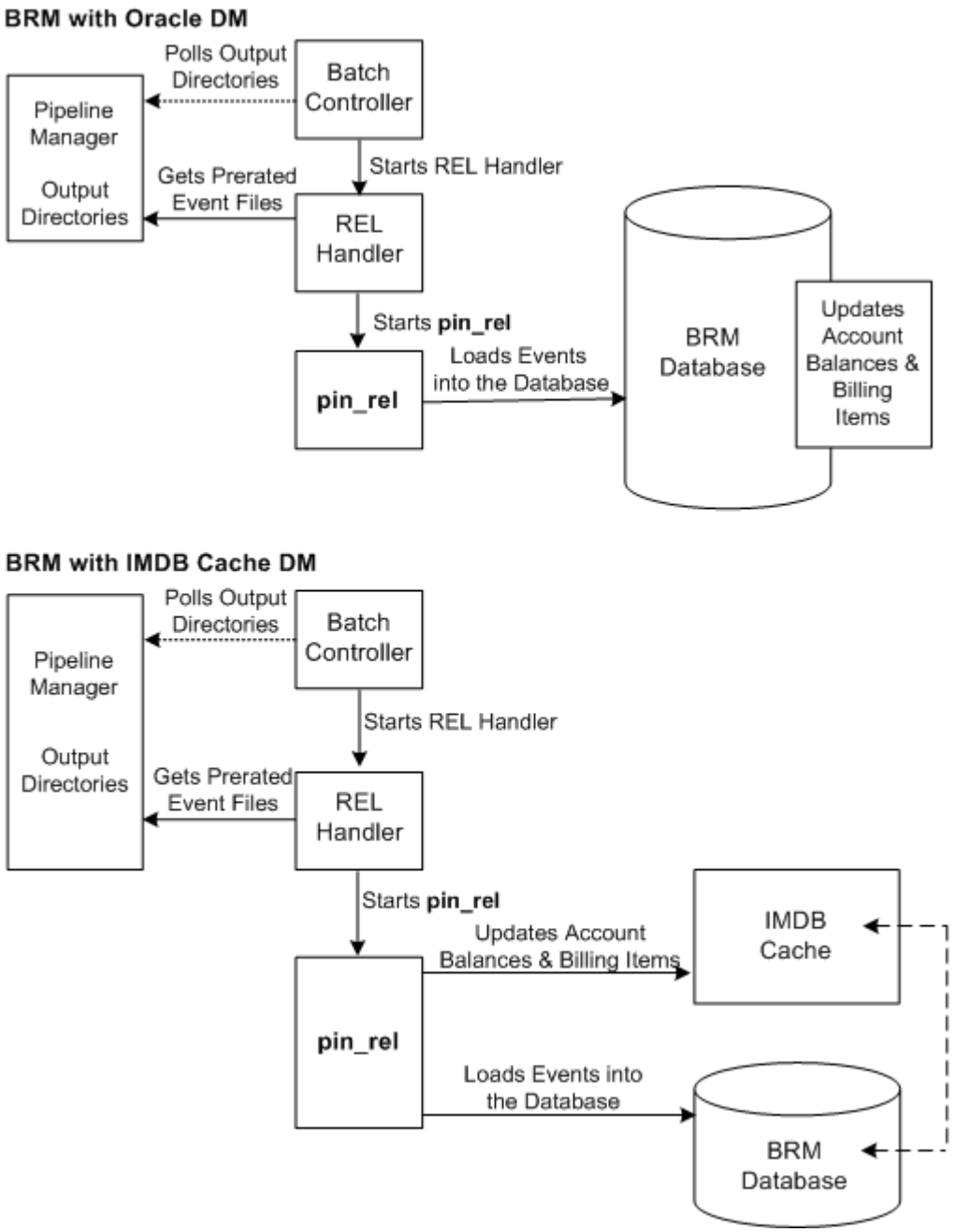
About Running RE Loader Automatically

To schedule RE Loader to run automatically, you must set up the following components:

- **Batch Controller**, which detects when an EDR file is present in the pipeline output directory and starts the RE Loader batch handler.
- **RE Loader batch handler**, which moves the EDR file from the pipeline output directory to the RE Loader processing directory and starts the RE Loader utility (**pin_rel**).

Figure 54-2 shows the RE Loader work flow when you schedule loading of events:

Figure 54-2 Work Flow of Scheduled Execution of RE Loader



The following actions are performed when RE Loader is scheduled to run automatically:

1. Batch Controller detects an EDR file in a pipeline output directory and starts the RE Loader batch handler.
2. The RE Loader batch handler moves the EDR file from the pipeline output directory to the RE Loader processing directory and starts the RE Loader utility (**pin_rel**).
3. RE Loader processes the file, loads the events into the BRM database, and updates the account balances, billing items, and journals. For more information, see "RE Loader Process Overview".
4. RE Loader batch handler moves the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded.

> **Note:**
>
> The RE Loader batch handler loads one file at a time. You can load more than one file at a time by configuring Batch Controller to call several RE Loader batch handler processes. For more information, see "About Running Multiple RE Loader Processes".

About Running the RE Loader Daemon

When you run the RE Loader daemon, the daemon creates RE Loader threads to load EDR files. Each RE Loader thread loads one EDR file. The number of threads that the RE Loader daemon creates depends on the maximum number of threads that you configure to run simultaneously at a particular time. Because you can configure multiple threads that can run at a time, you can load more than one EDR file simultaneously. This helps in increasing loading performance.

For example, if you configure 10 threads to run simultaneously at peak time, the RE Loader daemon creates 10 threads, which means that 10 EDR files are loaded at the same time.

> **Note:**
>
> Running the RE Loader daemon is the preferable method if most of the EDR files that you want to load are of small size.

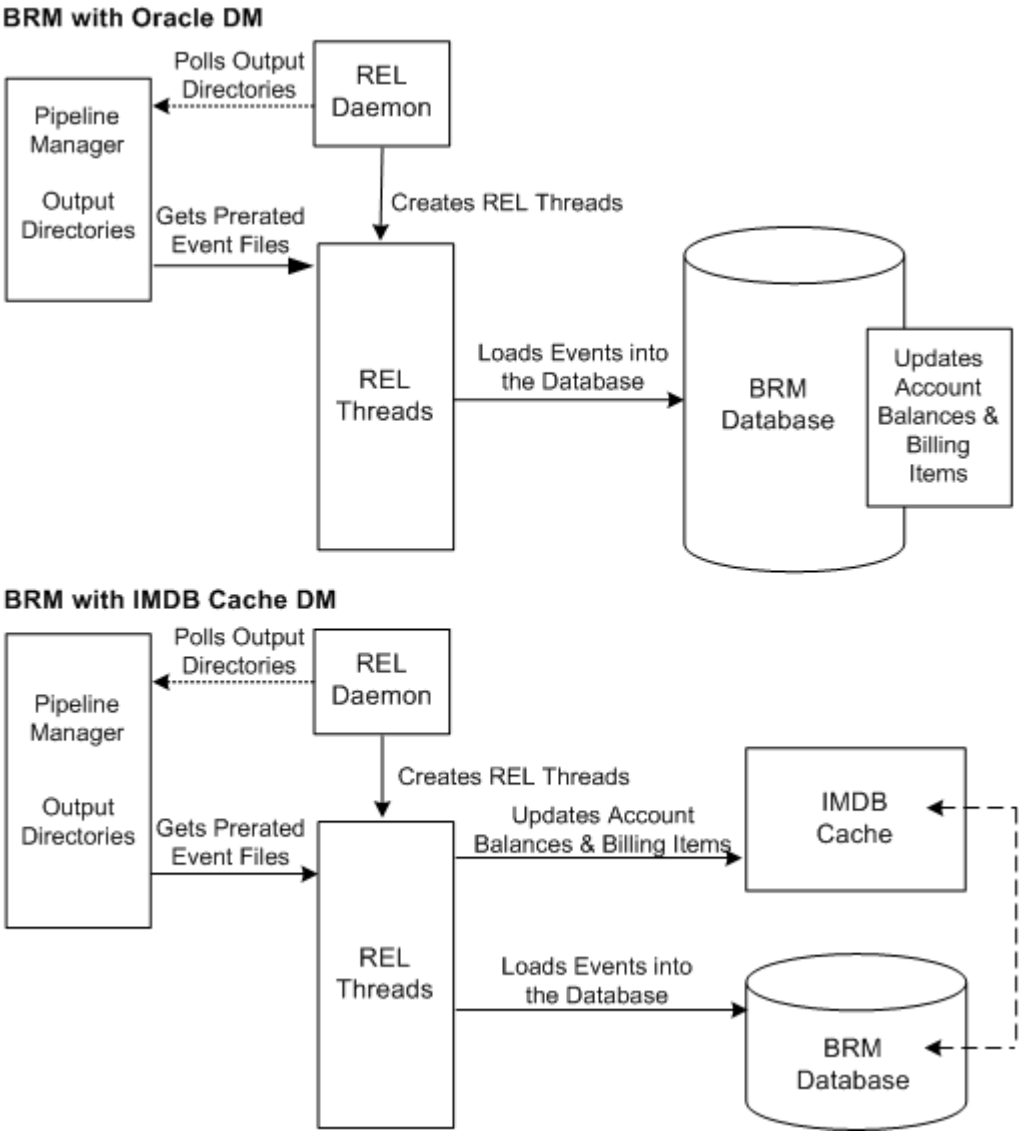
To run the RE Loader daemon, you must set up the following components:

- The RE Loader **Infranet.properties** file, which contains the entries for running the RE Loader daemon. See Table 56-5 for more information.
- The RE Loader daemon start and stop scripts (**start_rel_daemon** and **stop_rel_daemon**). See "Running the RE Loader Daemon" for more information.

You can also run the RE Loader daemon by using the **pin_ctl** utility.

Figure 54-3 shows the RE Loader work flow when you run the RE Loader daemon:

Figure 54-3 Work Flow of RE Loader Daemon



The following actions are performed when you run the RE Loader daemon:

1. The RE Loader daemon detects an EDR file in a pipeline output directory.
2. The RE Loader daemon creates RE Loader threads up to the maximum number of threads configured.
3. The RE Loader threads process the files, load the events into the BRM database, and update the account balances, billing items, and journals.
4. The RE Loader threads move the original file to an archive directory if the records are successfully loaded or to a reject directory if the records are not successfully loaded. These directories are configured in the RE Loader **Infranet.properties** file.

About Running Multiple RE Loader Processes

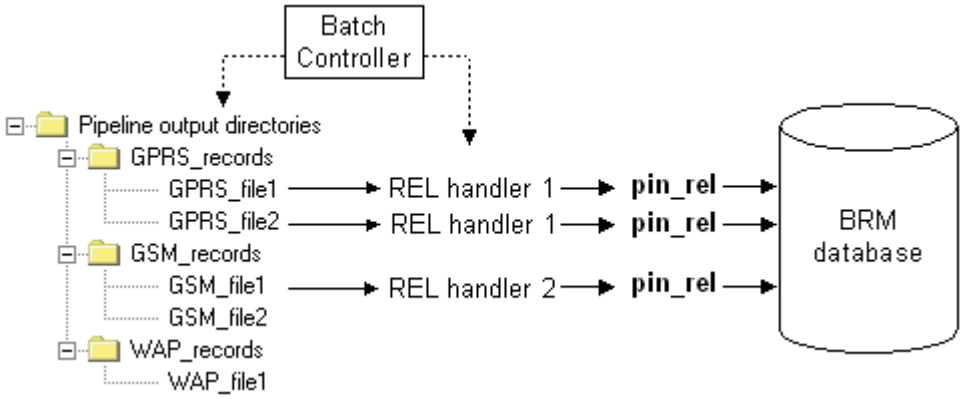
To achieve better loading performance, the sample Batch Controller configuration file (**SampleBatchControllerInfranet.properties**) is set to run three RE Loader processes in parallel. This means that when you schedule RE Loader to run automatically, Batch Controller starts up to three instances of the RE Loader batch handler. Each instance of the RE Loader batch handler starts an instance of the RE Loader utility.

If you configure RE Loader to run manually, you can start multiple processes from the command line.

> **Note:**
>
> If you run multiple RE Loader processes in parallel, configure Pipeline Manager to delete empty output streams. For more information, see "Configuring Pipeline Manager to Delete Empty Output Streams".

Figure 54-4 shows an example of three RE Loader batch handler and **pin_rel** utility processes running to load EDR files. Each utility loads one file into the BRM database:

Figure 54-4 Example of Three Batch Handlers and pin_rel Utilities



The following actions are performed when multiple RE Loader processes are configured:

1. Batch Controller starts an RE Loader batch handler for each new file it detects in the pipeline output directory, up to the maximum number of RE Loader batch handler processes configured.
2. Each RE Loader batch handler process starts an RE Loader utility process.
3. The RE Loader utility processes the events in the file in three phases:
   - Preprocessing
   - Loading
   - Account balance, bill item, and journal updates

   During the loading phase, the database tables are locked so that only one RE Loader process can load events at one time.

4. Each RE Loader process polls the other processes to see whether they are currently in the loading phase and waits its turn to load events. When the first process is loading, the second process performs prepreprocessing tasks while waiting its turn. When the first process finishes loading, the second process loads while the first process performs account balance, bill item, and journal updates for the events it loaded:

Process 1	Preprocessing	Loading	Updating	Preprocessing	. . .
Process 2	Not started	Preprocessing	Loading	Updating	. . .
Process 3	Not started	Preprocessing	Waiting	Loading	. . .

5. When an RE Loader process has completed all three phases, it is free to process another file.

The files are loaded in sequence, one directory at a time. If the number of RE Loader processes is set to 3, only three handler processes can run at one time for all directories.

Setting the Optimal Number of RE Loader Processes

Because there are three phases to processing EDR files, the optimal number of RE Loader processes to configure is at least three. If you want to configure more than three processes, you should test your RE Loader performance. Because only one RE Loader process can load events at a time, having more than three means those that have finished the preprocessing phase wait their turn to load events. Depending on the size of your EDR files and the time it takes to load events, configuring four or five processes might save time.

You configure the number of RE Loader processes to run in parallel by setting the number of RE Loader batch handlers to start in the Batch Controller configuration file. For more information, see "Configuring Batch Controller".

Configuring Pipeline Manager to Delete Empty Output Streams

If you run multiple RE Loader processes, you should configure Pipeline Manager to delete empty output streams.

Pipeline Manager generates files based on the number of output streams that are running. If some of the output streams are empty, the pipeline can produce empty files, which causes an error when RE Loader attempts to load the empty files.

To produce only one output stream, set the **DeleteEmptyStream** pipeline registry entry to **True**. This is the default. For more information, see "Configuring Output for Rated Events" and "OUT_GenericStream".

About Backing Up RE Loader Files

By default, RE Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down ungracefully.

To prevent data loss when your system shuts down:

- Make full backups of the BRM database on a regular basis.
- Archive all successfully loaded files until you make a full database backup.

You can re-enable redo generation, at the cost of loading performance, by modifying the RE Loader control files. For information, see "Configuring Whether to Perform Redo Generation".

About Handling Errors

If any errors occur during event loading, all events loaded in that session are deleted from the database. After events have been successfully loaded, if any errors occur during the update procedure, you can correct the errors and then update the relevant events by rerunning the RE Loader utility. The utility detects that the events loaded correctly and performs only the update procedure. For more information, see "Troubleshooting Event Loading".

About Using RE Loader in a Multischema System

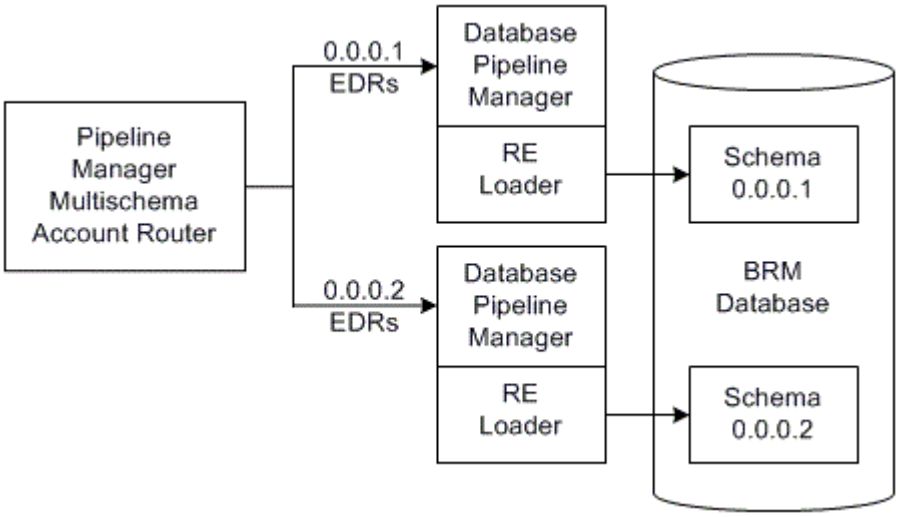
If you use a multischema system, you must set up the following for each BRM database schema in your system:

- Pipeline Manager instance. Each instance of Pipeline Manager must also have its own set of output files.
- RE Loader instance. Each instance of RE Loader must also have its own set of RE Loader processing directories.
- (Running RE Loader automatically only) An instance of Batch Controller and the RE Loader batch handler.

You must also install and configure a separate instance of Pipeline Manager. This Pipeline Manager multischema account router routes EDRs to the appropriate schema Pipeline Manager instance based on the account's database number.

Figure 54-5 shows the flow of data in a multischema system:

Figure 54-5 Multischema System Data Flow



55

Installing Rated Event Loader

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader software.

About Configuring RE Loader

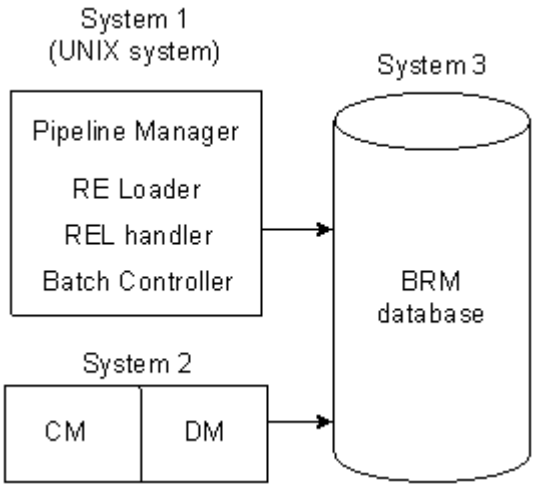
RE Loader uses Batch Controller, which needs access to the pipeline output files. Therefore, Pipeline Manager, Batch Controller, and RE Loader software should be installed on the same system that contains the pipeline output files.

If Pipeline Manager and RE Loader are on different systems, you need to map the Pipeline Manager output directories to a drive local to RE Loader.

Figure 55-1 shows the recommended configuration for installing RE Loader and its related features:

> **Note:**
>
> Event Extraction Tool can be installed on any of these systems or on its own system.

Figure 55-1 Recommended Configuration for RE Loader Installation



> **Note:**
>
> Though it is possible to install RE Loader on a BRM system or the database system, you will get better performance if you install it on the Pipeline Manager system.

Installing RE Loader

To install RE Loader, perform the procedures in these sections:

1. Granting Execute Permission for dbms_lock
2. Granting Write Permission to the DM
3. Installing the RE Loader Package
4. Creating Your RE Loader Database Partitions
5. Returning DM Permissions to their Original Values

Granting Execute Permission for dbms_lock

Before you install RE Loader, you must grant execute permission to *pin_user* for **dbms_lock**:

1. Log in to your database as the SYS user:

```
% sqlplus sys@databaseAlias
Enter password: password
```

2. Grant execute privileges to *pin_user*:

```
SQL> grant execute on dbms_lock to pin_user
```

Granting Write Permission to the DM

When you install RE Loader on a system where BRM is not installed, you must grant the DM write permission before installing RE Loader.

Perform the following on all machines containing a DM:

1. In a text editor, open your DM configuration file: *BRM_home***/sys/dm_oracle/pin.conf**.

   BRM_home is the directory where you installed BRM components.

2. Write down the values of your **dd_write_enable_fields**, **dd_write_enable_objects**, **dd_write_enable_portal_objects**, and **dd_mark_as_portal** entries.

3. Set the values of the following entries to **1**:

```
- dm dd_write_enable_fields 1
- dm dd_write_enable_objects 1
- dm dd_write_enable_portal_objects 1
- dm dd_mark_as_portal 1
```

> **Note:**
>
> If any entry is not in the file, add it.

   For more information, see comments in the DM **pin.conf** file.

4. Save and close the file.

5. Stop and restart the DM.

   You can now install RE Loader.

Installing the RE Loader Package

> **Note:**
>
> If you are upgrading from a previous version of RE Loader, you must make sure that all available unrated events are rated by Pipeline Manager and loaded before installing this version of RE Loader.

To install RE Loader, see "Installing Individual BRM Components" in *BRM Installation Guide*.

Creating Your RE Loader Database Partitions

> **Note:**
>
> You must perform this step to ensure that the new event tables have the same partitioning layout as your existing event tables. If you install several optional components, perform this step only after installing the last component.

To create partitions for RE Loader events:

1. On the system where BRM is installed, go to the *BRM_home***/apps/partition_utils** directory.
2. Run the **partition_utils** utility to enable delayed-event partitions:

   ```
   perl partition_utils.pl -o enable -t delayed -c storable_class
   ```

   where *storable_class* specifies the event classes for which you want partitioning.

   > **Note:**
   >
   > You must create partitions for all subclasses of a specific service event type that you want to load.

   For example, this command creates partitions for **/event/delayed/session/telco/gsm** delayed events:

   ```
   perl partition_utils.pl -o enable -t delayed -c /event/session/telco/gsm
   ```

Your RE Loader installation is now complete.

Returning DM Permissions to their Original Values

To return your DM permissions to their original values:

1. In a text editor, open your DM configuration file: *BRM_home***/sys/dm_oracle/pin.conf**.
2. Restore the following entries to their original values (the values they had before you modified them). The default value for each entry is **0**:

```
- dm dd_write_enable_fields
- dm dd_write_enable_objects
- dm dd_write_enable_portal_objects
- dm dd_mark_as_portal
```

3. Save and close the file.
4. Stop and restart the DM.

What's Next?

Configure RE Loader. See "Configuring Rated Event Loader".

Uninstalling RE Loader

To uninstall RE Loader, see "Uninstalling Optional Components" in *BRM Installation Guide*.

56
Configuring Rated Event Loader

This chapter describes how to set up Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader to load events that have been rated by Pipeline Manager.

For an overview of loading pipeline events, see "Understanding Rated Event Loader".

Setting Up Your System for RE Loader

To set up your system for RE Loader:

1. Install RE Loader. Install one instance of RE Loader for each BRM database schema.
2. Configure your system to use the correct Oracle library files. See "Configuring Oracle Libraries for RE Loader".
3. Configure the RE Loader configuration (**Infranet.properties**) file. See "Configuring the RE Loader Infranet.properties File".
4. Create RE Loader processing directories. See "Setting Up RE Loader Processing Directories".
5. Configure RE Loader to work with multischema systems. See "Setting Up RE Loader for Multischema Systems".
6. Configure Batch Controller to run RE Loader automatically. See "Configuring RE Loader to Run Automatically".
7. Disable invoice event caching. See "Disabling Invoice Event Caching".
8. Set up delayed billing. See "Enabling a Billing Delay for CDRs".
9. Increase the maximum field length in input data files. See "Configuring Field Lengths for Input Data Files".

Configuring Oracle Libraries for RE Loader

RE Loader requires Oracle 32-bit libraries, and Pipeline Manager requires Oracle 64-bit libraries. If RE Loader and Pipeline Manager reside on the same system, make sure both the 32-bit and 64-bit Oracle libraries are installed on your system.

To support the libraries on the same machine, set up your Oracle environment so that RE Loader points to the 32-bit Oracle libraries and Pipeline Manager points to the 64-bit Oracle libraries:

1. Install the 32-bit and 64-bit libraries in separate directories on the Pipeline Manager system.
2. Set environment variables in the **pin_rel** and **SampleRelHandler_config.values** files to point to the 32-bit libraries. See "Setting the Oracle Library Paths".
3. Set up your default Pipeline Manager environment in the *Oracle_home*/**.cshrc** file to use Oracle 64-bit libraries. See Example 56-1.

Setting the Oracle Library Paths

To set the Oracle library paths:

1. Open the *BRM_home*/**apps/pin_rel/pin_rel** file in a text editor. *BRM_home* is the directory where you installed BRM components.
2. Add the following *before* the command that runs Java.

```
setenv ORACLE_HOME Oracle_home
setenv SHLIB_PATH ${ORACLE_HOME}/lib
setenv PATH ${PATH}:${ORACLE_HOME}/bin
```

> **Note:**
>
> For Oracle 11*g* R2 installations, *Oracle_home* points to the location where you installed the full 32-bit client software.

3. Save and close the file.
4. Open the *BRM_home*/**apps/pin_rel/SampleRelHandler_config.values** file in a text editor.
5. Add these lines between the FILETYPE and HANDLER_DIR variables:

> **Note:**
>
> The paths you set depend on your system setup.

```
$ENV{'ORACLE_HOME'} = "/u01/app/oracle/product/11i64";
$ENV{'SHLIB_PATH'} = "/usr/lib:/opt/portal/7.5/lib:/u01/app/oracle/product/11i64/lib";
$ENV{'PATH'} = ".:/bin:/usr/bin:/usr/local/bin:/u01/app/oracle/product/11i64/bin:/opt/
portal/7.5/bin";
```

6. Save and close the file.

Example 56-1 Example of the Oracle 64-Bit Setting in the .cshrc File

```
setenv ORACLE_HOME /u01/app/oracle/product/10g64
setenv ORACLE_SID       PIND10g64
setenv NLS_LANG American_America.AL32UTF8
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$ORACLE_HOME/lib32:$ORACLE_HOME/rdbms/
lib32
setenv LD_LIBRARY_PATH_64 $IFW_HOME/lib:$ORACLE_HOME/lib:$ORACLE_HOME/
rdbms/lib
setenv ORACLE_BIN $ORACLE_HOME/bin
setenv ORACLE_DOC $ORACLE_HOME/odoc
setenv ORA_NLS33 $ORACLE_HOME/ocommon/nls/admin/data
alias ora 'cd $ORACLE_HOME'
set path = ($path $ORACLE_BIN)
```

Configuring the RE Loader Infranet.properties File

The **Infranet.properties** file contains configuration information for processing event data record (EDR) files, such as the location of the RE Loader processing directory, how to connect to the BRM database, and how to process specific event types. The RE Loader **Infranet.properties** file contains the following sections:

- Connection entries. This section specifies how to connect to your database.
- General processing entries. This section defines your RE Loader log file, how to connect to BRM and the BRM database, and the fields to process in your EDR files.
- Default processing entries. This section defines how to process any event type. You can override these values for specific events by using the storable class-specific entries.
- Storable class-specific processing entries. This section defines how to process specific event types (for example, **/event/delayed/session/gprs** events or **/event/delayed/session/telco/gsm** events). All configuration information in this section overrides your default entries.
- The RE Loader daemon entries. This section specifies how to run the RE Loader daemon. If you do not run the RE Loader daemon, you can ignore the entries in this section.

To configure your RE Loader **Infranet.properties** file:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.
2. Specify how to connect to your database by setting the entries shown in Table 56-1. Replace *LogicalPartID* with the logical partition number, such as **0.1.0.1**.

Table 56-1 Database Connection Entries

Entry	Description
infranet.rel.dbtype	Specifies the BRM database type. The default is **oracle**.
infranet.rel.dbname	Specifies the BRM database name. **Note**: Your database name is the TNSNAMES alias in the *Oracle_home***/network/admin/tnsnames.ora** file. The default is **pindb**.
infranet.rel.userid	Specifies the user ID for connecting to the BRM database. The default is **pin**.
infranet.rel.password	Specifies the password for connecting to the BRM database.

3. Set the general processing entries shown in Table 56-2.

Table 56-2 General Processing Entries

Entry	Description
infranet.log.file	Specifies the name of the RE Loader log file. The default is **rel.pinlog**.
infranet.log.name	Specifies the name of the application. The default is **REL** for RE Loader.
infranet.log.level	Specifies the log reporting level: • **1** specifies error-level reporting. • **2** specifies warning-level reporting. • **3** specifies debug-level reporting. The default is **1**.

Table 56-2 (Cont.) General Processing Entries

Entry	Description
infranet.log.logallebuf	Specifies whether RE Loader automatically logs all EbufExceptions. The default is **True**.
infranet.rel.use_end_time	Specifies whether RE Loader uses the start time or end time of the rated event for deciding the billing cycle. • **1** specifies that RE Loader uses the end time of the rated event for deciding the billing cycle. The default is **1**. • **0** specifies that RE Loader uses the start time of the rated event for deciding the billing cycle.
infranet.connection	Specifies the user login name. For example: `infranet.connection=pcp://root.0.0.0.1:`*`password`*`@localhost:11960/service/pcm_client` RE Loader uses this Connection Manager (CM) connection to log audit information. **Important:** RE Loader writes audit information to the database specified in this entry. If you use a multischema system, you might want to modify this entry to write audit information to the schema where the records are loaded.
infranet.login.type	Specifies whether RE Loader requires a login name and password to log in to BRM. • **0** specifies that a login name and password *are not* required. • **1** specifies that a login name and password *are* required. The default is **1**.
infranet.rel.dbhost	Specifies the database machine's host name.
infranet.rel.dbport	Specifies the database port number. The default is **1433**.
infranet.failover	In high-availability systems, specifies the secondary CM connection. For example: `infranet.failover.1 = pcp://root.0.0.0.`*`db_no`*`:`*`password`*`@`*`failover_host`*`:`*`failover_port`*`/service/pcm_client`
infranet.rel.polling_interval	Specifies the interval, in milliseconds, that RE Loader checks the database to see whether another process is loading. The default is **1000**. The polling interval depends on the number and size of your input files. If you have very large files, make the polling interval longer. If you have many small files, make the interval shorter.
infranet.rel.polling_time_out	Specifies the time, in milliseconds, that RE Loader waits to load events before exiting. The default is **600000**. The time-out period depends on the number and size of your input files and how many parallel RE Loader processes are running. If you have very large files or many processes, make the time-out period longer.

Table 56-2 (Cont.) General Processing Entries

Entry	Description
infranet.rel.partition_set_number	Specifies the partition set number, from 1 through 7. This entry applies only to BRM databases with multiple delayed partition sets. The default is **1**. • **1** uses delayed partition set P_1D to P_12D. • **2** uses delayed partition set P_1D to P_12D2. • **3** uses delayed partition set P_1D to P_12D3. • **4** uses delayed partition set P_1D to P_12D4. • **5** uses delayed partition set P_1D to P_12D5. • **6** uses delayed partition set P_1D to P_12D6. • **7** uses delayed partition set P_1D to P_12D7.
infranet.rel.updater_threads	Specifies the number of threads dedicated to the update and preupdate stored procedures. You can specify a fixed number of threads or configure RE Loader to adjust the number of threads based on the number of database objects to update. To specify a fixed number of threads, set the entry equal to the desired number of threads. To configure RE Loader to automatically adjust the number of threads, set the entry to **0**. RE Loader spawns the number of threads shown below: Less than 1,000 objects: 2 threads Between 1,000 and 200,000 objects: 4 threads More than 200,000 objects: 8 threads The default is **4**. **Note:** Specifying a number of threads that exceeds the number of CPUs in your system may cause deadlock due to a lack of system resources. If you set the **infranet.rel.updater_threads** entry to a value greater than 8, RE Loader returns a warning message and continues processing.
infranet.rel.validate_dbnumber	Specifies whether RE Loader performs an extra validation step to ensure that it is loading a call detail record (CDR) file into the correct database schema. The default is **True**. **Important:** Use this option only for debugging. In a production environment, set this to **False**. Setting it to **True** degrades performance while loading data into the database. For more information, see "Turning Off Database Verification to Improve Processing Performance".
infranet.rel.validate_indexes	Specifies whether RE Loader verifies that the database indexes are correct before loading data into the database. The default is **False**. **Important:** Use this option only for debugging. In a production environment, set this to **False**. Setting it to **True** degrades performance while loading data into the database.For more information, see "Turning Off Index Verification to Improve Database Loading Performance".
infranet.rel.max_increment_by	Specifies the number of database schemas in your system. This value is used by the POID generation algorithm to ensure that POIDs are unique across all databases schema in your system. The default is **20**. For more information, see "Preventing POID Errors in Multischema Systems".

Table 56-2 (Cont.) General Processing Entries

Entry	Description
infranet.rel.sort.limit	Defines the maximum number of CDRs that the preprocessing script can sort by account ID. This improves performance later during the balance updating process. If the number of CDRs in the input file is greater than the **infranet.rel.sort.limit** value, the preprocessing script does not sort the CDRs. The default is **100000**.
infranet.rel.custom_error_codes	Specifies the name of the custom error code file. The default name is **CustomErrorCodes.properties**. If you want to move this file from its default location, you must create a symbolic link between the name of the file and its new location. To create this link, go to the *BRM_home*/**apps/pin_rel** directory and enter the following at the command prompt: `$ ln -s path_to_where_file_was_moved / CustomErrorCodes.properties ./CustomErrorCodes.properties`
infranet.rel.default.header.record_type	Specifies the header record type. The default is **010**.
infranet.rel.default.detail.record_type	Specifies the detail record type. The default is **020**.
infranet.rel.default.trailer.record_type	Specifies the trailer record type. The default is **090**.
infranet.rel.field.delimiter	Specifies the delimiter symbol. The default is **\t**, for tabs.
infranet.rel.header.position.storable_class	Specifies which field in the EDR file contains the storable class name. The default is **20**. **Note:** When you set this field to **0**, RE Loader uses the default storable class specified in **infranet.rel.default.storable_class**.
infranet.rel.header.position.creation_process	Specifies which field in the EDR file contains the name of the creation process (for example, whether the file contains prerated, rerated, or discount events). The default is **18**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.sender	Specifies which field in the EDR file contains the sender. The default is **3**.
infranet.rel.header.position.recipient	Specifies which field in the EDR file contains the recipient. The default is **4**.
infranet.rel.header.position.file_sequence	Specifies which field in the EDR file contains the file sequence number. The default is **5**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.creation_timestamp	Specifies which field in the EDR file contains the creation timestamp. The default is **7**. **Note:** You can specify **0** if you do not need this field validated.
infranet.rel.header.position.object_cache_type	Set this value to **0**.
infranet.rel.trailer.position.record_count	Specifies the field position of the field that contains the total number of detail records in the output file. The default is **7**. The field position starts with 1.
infranet.rel.file_extension.disc.transform_script	By default, this entry is commented out.
infranet.rel.file_extension.disc.transform_flags	By default, this entry is commented out.

4. Set the default configuration entries shown in Table 56-3. The configuration information in this section applies to all event types except for those defined in the storable class-specific section.

Table 56-3 Default Configuration Entries

Entry	Description
infranet.rel.default.interim_directory	Specifies the RE Loader processing directory. This is the location where preprocessed events are temporarily stored before they are loaded. The default is *BRM_home*/**apps/pin_rel**.
infranet.rel.default.supported_creation_processes	Specifies which creation processes are supported. By default, RE Loader supports all creation processes: • **RATING_PIPELINE** specifies that the file was last processed by the rating pipeline and therefore contains *prerated* events. • **RERATING_PIPELINE** specifies that the file was last processed by the rerating pipeline and therefore contains *rerated* events. • **PIN_REL_TRANSFORM_CDR** specifies that the file was last processed by the **pin_rel_transform_cdr.pl** script and therefore contains *discount* events. • **SUSPENSE_CREATE** specifies that the RE Loader process will create new suspense records in the suspended usage table. • **SUSPENSE_UPDATE** specifies that the RE Loader process will update existing suspense records in the suspended usage table.
infranet.rel.default.failure_script	Specifies the script called when RE Loader attempts to reload events that previously failed to load into the database. The default is **pin_rel_handle_interim_files.pl**.
infranet.rel.default.failure_flags	Specifies the flag passed to the failure script. You can specify the following flags in the default **pin_rel_handle_interim_files.pl** failure script: • **0** to do nothing. • **1** to rename the interim files by appending **.saved.timestamp** to the file name. • **2** to delete the temporary files. The default is **1**.
infranet.rel.default.preprocess_script	Specifies the name of the preprocessing script. The default is **pin_rel_preprocess_cdr.pl**.
infranet.rel.default.preprocess_flags	Specifies the flag passed to the preprocessing script. The default is **0**.

Table 56-3 (Cont.) Default Configuration Entries

Entry	Description
infranet.rel.default.load_util	Specifies the name of the load utility. For Oracle's SQL Loader, it also specifies whether the utility uses direct-path loading or conventional-path loading: • **Direct-path loading**. This is the fastest way to load events into the database. It can be 10% to 30% faster than conventional-path loading, depending on the file size, memory size, storage configuration, and storage performance. However, direct-path loading has limits for concurrent system activities. When an event is loaded in direct-path mode, the load utility locks the event's entire partition and some of the table's indexes. This prevents other operations from updating or reading the event table. Direct-path mode is recommended when the event table will have limited concurrent usage. • **Conventional-path loading**. This is the recommended loading mode if BRM will perform many concurrent operations on the event table. For example, use conventional-path loading if BRM is rerating events, performing billing-time taxation, or generating detailed invoices concurrently with RE Loader. Conventional mode is also recommended if you have small source files for RE Loader because the performance gained by using direct-path loading is surpassed by the mode's preprocessing and file-handling overhead. **Important:** If you use conventional-path loading, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option. To specify the load utility name and loading mode: • *UtilityName* **direct=true unrecoverable** specifies to use direct-path loading. This is the default. • *UtilityName* **direct=false** specifies to use conventional-path loading. The default is **sqlldr direct=true streamsize=5000000 readsize=10000000 unrecoverable**.
infranet.rel.default.preupdater_sproc	Specifies the name of the preupdate stored procedure. The default is **pin_rel.pin_rel_pre_updater_sp**.
infranet.rel.default.preupdater_batch_size	Specifies the size of the preupdate batch. The default is **5**.
infranet.rel.default.preupdater_flags	Specifies the flag passed to the preupdate stored procedure. The default is **1**.
infranet.rel.default.updater_sproc	Specifies the name of the update stored procedure. The default is **pin_rel.pin_rel_updater_sp**.
infranet.rel.default.updater_batch_size	Specifies the size of the update batch. The default is **5**.
infranet.rel.default.updater_flags	Specifies the flag passed to the update stored procedure. The default is **1**.
infranet.rel.default.success_script	Specifies the script called when RE Loader successfully loads a batch of events into the BRM database. The default is **pin_rel_handle_interim_files.pl**.
infranet.rel.default.success_flags	Specifies the flag passed to the success script. You can specify the following flags in the default **pin_rel_handle_interim_files.pl** script: • **0** to do nothing. • **1** to rename the interim files by appending **.saved.timestamp** to the file name. • **2** to delete the temporary files. The default is **1**.

Table 56-3 (Cont.) Default Configuration Entries

Entry	Description
infranet.rel.default.storable_class	Specifies the storable class you are loading. The default is **/event/delayed/session/gprs**. **Important**: If you use conventional-path loading, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option.
infranet.rel.default.creation_process	Specifies whether the file contains prerated, rerated, or discount events: • **RATING_PIPELINE** specifies that the file was last processed by the rating pipeline and therefore contains *prerated* events. • **RERATING_PIPELINE** specifies that the file was last processed by the rerating pipeline and therefore contains *rerated* events. • **PIN_REL_TRANSFORM_CDR** specifies that the file was last processed by the **pin_rel_transform_cdr.pl** script and therefore contains *discount* events. **Important:** RE Loader can dynamically source the creation process from the EDR header file. Uncomment this entry only if all of your EDR files come from the same creation process. The default is **RATING_PIPELINE**.
infranet.rel.ece_preprocessed	Always **FALSE** for Pipeline Manager.

5. If necessary, set the storable class-specific entries shown in Table 56-4. These settings override the default settings for the specified storable class.

> **Note:**
>
> For each storable class, only the **infranet.rel.storable_class.***classname***.number_of_tables** and **infranet.rel.storable_class.***classname***.table.***N***.name** entries are mandatory. RE Loader uses the default settings for any undefined storable class-specific entries.

When editing these entries:

- Create a set of entries for each event type you want to load.
- Replace *classname* with the appropriate storable class name. For example, use **event_delayed_session_gprs** for the **/event/delayed/session/gprs** storable class.
- Create a set of ***.table.***N***.*** entries for each table. For example, if the storable class contains three tables, create a set of ***.table.1.*** entries, a set of ***.table.2.*** entries, and a set of ***.table.3.*** entries.

Table 56-4 Storable Class-Specific Configuration Entries

Entry	Description
infranet.rel.storable_class.*classname***.interim_directory**	RE Loader processing directory. This is the location where preprocessed events are temporarily stored before they are loaded.
infranet.rel.storable_class.*classname***.supported_creation_processes**	Specifies whether the file contains prerated, rerated, or discount events.
infranet.rel.storable_class.*classname***.failure_script**	Specifies the script to call when RE Loader attempts to load events that previously failed to load into the database.

Table 56-4 (Cont.) Storable Class-Specific Configuration Entries

Entry	Description
infranet.rel.storable_class.*classname*.failure_flags	Specifies the flag to pass to the failure script.
infranet.rel.storable_class.*classname*.preprocess_script	Specifies the name of the preprocessing script.
infranet.rel.storable_class.*classname*.preprocess_flags	Specifies the flag to pass to the preprocessing script.
infranet.rel.storable_class.*classname*.number_of_tables	Specifies the number of tables in the storable class. **Important**: This entry is mandatory for each event type.
infranet.rel.storable_class.*classname*.table.*N*.name	Specifies the name of a storable class table. **Important**: This entry is mandatory for each event type.
infranet.rel.storable_class.*classname*.table.*N*.load_util	Specifies the name of the load utility. For Oracle's SQL Loader, it also specifies whether the utility uses direct-path loading or conventional-path loading: – *UtilityName* **direct=true unrecoverable** specifies to use direct-path loading. – *UtilityName* **direct=false** specifies to use conventional-path loading. **Important:** If you use conventional-path loading, use the APPEND option in your RE Loader control files. Do not use the TRUNCATE option.
infranet.rel.storable_class.*classname*.table.*N*.control_file	Specifies the control file to use when loading the data file into the database.
infranet.rel.storable_class.*classname*.preupdater_sproc	Specifies the name of the preupdater stored procedure.
infranet.rel.storable_class.*classname*.preupdater_batch_size	Specifies the preupdater batch size.
infranet.rel.storable_class.*classname*.preupdater_flags	Specifies the flag to pass to the preupdater stored procedure.
infranet.rel.storable_class.*classname*.updater_sproc	Specifies the name of the updater stored procedure.
infranet.rel.storable_class.*classname*.updater_batch_size	Specifies the updater batch size.
infranet.rel.storable_class.*classname*.updater_flags	Specifies the flag to pass to the updater stored procedure.
infranet.rel.storable_class.*classname*.success_script	Specifies the script to call when RE Loader successfully loads a data file into the BRM database.
infranet.rel.storable_class.*classname*.success_flags	Specifies the flag to pass to the success script when RE Loader successfully loads a data file into the BRM database.

6. (Optional) To run the RE Loader daemon, add or modify the RE Loader daemon entries shown in Table 56-5.

   When editing these entries, replace *event* with the appropriate event type.

Table 56-5 RE Loader Daemon Entries

Entry	Description
batch.check.interval	Specifies the time interval, in seconds, to monitor files from the pipeline output directory. The default is **5**.
batch.file.rename.extension	Specifies the file name extension that the RE Loader daemon uses to rename the interim files before processing them. The default is **.bc**.

Table 56-5 (Cont.) RE Loader Daemon Entries

Entry	Description
batch.start.highload.time	Specifies the start time of your system's busiest period. Specify the hour, minute, and second, in *hhmmss* format, using the 24-hour clock.
batch.end.highload.time	Specifies the start time of your system's slowest period. Specify the hour, minute, and second, in *hhmmss* format, using the 24-hour clock.
batch.lock.socket.addr	Specifies the port address of the process.
batch.rel.archiveDir	Specifies the full path to the directory where a successfully processed file is archived. This is the default archive directory for all the event handlers.
batch.rel.rejectDir	Specifies the full path to the directory where an unsuccessfully processed file is stored. This is the default reject directory for all the event handlers.
batch.random.events	Specifies the name of the event type to process. If you have two or more event types, separate each with a comma, but no blank space. For example, TEL,SMS,GPRS.
*event***.max.at.highload.time**	Specifies the highest number of the RE Loader threads that are permitted to run simultaneously for this event during the high-load time; that is, from **batch.start.highload.time** to **batch.end.highload.time**. For example, if *event***.max.at.highload.time** is **2**, two threads are permitted to run simultaneously for this event during the high-load time.
*event***.max.at.lowload.time**	Specifies the highest number of the RE Loader threads that are permitted to run simultaneously for this event during the low-load time; that is, from **batch.end.highload.time** to **batch.start.highload.time**. For example, if *event***.max.at.lowload.time** is **2**, two threads are permitted to run simultaneously for this event during the low-load time.
*event***.file.location**	Specifies the full path name of the directory to monitor for the arrival of new files that match the pattern in *event***.file.pattern**.
*event***.file.pattern**	Specifies the file name pattern to look for. You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported.
*event***.archiveDir**	(Optional) Specifies the full path to the directory where a successfully processed file is archived for a particular event handler. When multiple event handlers are configured, configure this entry to specify the directory that archives files from a particular event handler.
*event***.rejectDir**	(Optional) Specifies the full path to the directory where an unsuccessfully processed file is stored for a particular event handler. When multiple event handlers are configured, configure this entry to specify the directory that stores files from a particular event handler.
*event***.file.type**	Specifies the type of input CDR file. Always **STANDARD** for Pipeline Manager.

7. Save and close the file.

Setting Up RE Loader Processing Directories

The processing directory is where you run RE Loader. It must include all RE Loader execution scripts, configuration files, and RE handler files. Most systems require only one RE Loader processing directory, but you must create additional processing directories in the following situations:

- You have a multischema system. Each database schema in your system must have a corresponding instance of RE Loader and the RE Loader processing directory.

- You want to distribute load among multiple instances of RE Loader. If your system contains multiple pipelines that generate a large number of output files, you can increase database loading performance by using multiple RE Loader instances. In this case, each instance has its own processing directory and a corresponding pipeline output directory.

> **Note:**
>
> Do not configure multiple instances of RE Loader to process the same file type from the same directory. Doing so provides no advantage and can cause errors.

To set up processing directories, do the following in each instance of RE Loader:

1. In your *BRM_home***/apps/pin_rel** directory, create processing directories.

   For example, create a *BRM_home***/apps/pin_rel/GPRS** directory and a *BRM_home***/apps/pin_rel/GSM** directory.
2. Configure the **Infranet.properties** file. See "Configuring the RE Loader Infranet.properties File".
3. Copy all files from the *BRM_home***/apps/pin_rel** directory to each processing directory.
4. If RE Loader and Pipeline Manager are installed on separate systems, do the following:
   a. Go to the system where Pipeline Manager is installed.
   b. Mount the RE Loader processing directories onto the Pipeline Manager output directory.
5. Follow the instructions in "Configuring SE Loader for Standard Recycling".

Setting Up RE Loader for Multischema Systems

To set up RE Loader for multischema systems:

1. Install and configure one instance of Pipeline Manager for *each* BRM database schema and one instance for the multischema account router.

   For example, if your BRM system contains three schemas, install and configure four instances of Pipeline Manager.
2. In the multischema account router instance of Pipeline Manager, configure the following:
   a. Set the FCT_AccountRouter module's **Mode** registry entry to **ROUTER** and configure the module's **streams** registry entry to create an output stream for each schema Pipeline Manager instance. See "FCT_AccountRouter".
   b. Set the DAT_AccountBatch module's **UseAsRouter** registry entry to **True**. See "DAT_AccountBatch".

See "Using Pipeline Manager with Multiple Database Schemas" for more information.

Setting Up RE Loader for Virtual Column-Enabled Systems

This section explains the setup required for RE Loader to work in a virtual column-enabled system.

RE Loader populates some of the event tables. After you generate virtual columns on event tables in your BRM installation, you must run the **pin_gen_classid_values.pl** script. Running

the script ensures that the proper mapping of BRM object types and their corresponding object IDs is created for your extended event objects in a virtual column-enabled system.

To set up RE Loader for virtual column-enabled systems:

1. Go to *BRM_home*/**setup/scripts**.
2. Open the **pin_gen_classid_values.pl** file and verify that the first line in the file is pointing to the location of Perl in your installation.
3. Run the Perl script **pin_gen_classid_values.pl**.

   Running the script regenerates the **classid_values.txt** file that is used by RE Loader. The **classid_values.txt** file has the mapping of BRM object types (**poid_types**) and their corresponding object IDs (**object_ids**).

If you have extended BRM objects and these extended objects are new event subclasses that impact RE Loader, you must create new SQL Loader (**sqlldr**) control files. In virtual column-enabled systems, the RE Loader **sqlldr** control files must use the keywords VIRTUAL_CHAR and VIRTUAL_CONSTANT in the section that specifies the data definition of rows and also in the constant section.

Configuring RE Loader to Run Automatically

To configure RE Loader to run automatically, do the following for *each instance of RE Loader*:

- Configuring the RE Loader Batch Handler
- Configuring Batch Controller

> **Note:**
>
> For more information, see "About Running RE Loader Automatically".

- Specify each RE Loader batch handler and handler settings in the Batch Controller configuration file.

Configuring the RE Loader Batch Handler

> **Note:**
>
> If you use the ConfigurableValidityHandler batch handler for loading the validity periods of charge offers, discount offers, and balance elements that start on first usage, do not use the **SampleRelHandler_config.values** file as instructed below. Instead, configure the RE Loader batch handler in the ConfigurableValidityHandler configuration file (*BRM_home*/**apps/pin_rel/ConfigurableValidityHandler_config.values**). ConfigurableValidityHandler runs both the **pin_rel** utility and the utility for loading validity data. See "Configuring the ConfigurableValidityHandler Batch Handler".

To configure the RE Loader batch handler:

1. Give the **SampleRelHandler.pl** file a unique name. You will configure Batch Controller to call the handler using this name.

2. Create the following subdirectories: An **archive** subdirectory where successfully processed files can be stored and a **reject** subdirectory where unsuccessfully processed files can be stored.
3. Open the **SampleRelHandler_config.values** file and modify the entries shown in Table 56-6.

Table 56-6 Mandatory RE Loader Batch Handler Configuration Entries

Entry	Description
$FILETYPE	Specifies the EDR file-name pattern to look for. Change the value of this entry if you want to load only specific files. The default is ***.dat.bc** (any data file processed by Batch Controller). Batch Controller runs the RE Loader batch handler for each file with a name that matches this pattern. **Tip:** You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported.
$HANDLER_DIR	Specifies the full path to the directory containing the RE Loader batch handler, which is this processing directory.
$pinRELDir	Specifies the full path to the directory containing the RE Loader application, which is this processing directory.
$pinREL	Specifies the full path to the batch application executable.
$STAGING	Specifies the full path to the pipeline output directory. If you specify a directory other than the pipeline output directory, use the UNIX command that links the pipeline output directory to the input staging directory.
$PROCESSING	Specifies the full path to the directory from which EDR files are processed. The default is **$pinRELDir**. This must be the same directory specified in the following RE Loader **Infranet.properties** entries: • **infranet.rel.default.interim_directory** • **infranet.rel.storable_class.***classname***.interim_directory**
$ARCHIVE	Specifies the full path to the directory where a successfully processed file is archived. This is the **archive** subdirectory created in step 2. Change this value if you used a name other than the default, **$pinRELDir/archive**.
$REJECT	Specifies the full path to the directory where an unsuccessfully processed file is stored. This is the **reject** subdirectory created in step 2. Change this value if you used a name other than the default, **$pinRELDir/reject**.

4. Save and close the file.

Configuring Batch Controller

The RE Loader package includes Batch Controller. If your system already has Batch Controller installed, the RE Loader installer does not install another. Use the sample Batch Controller properties file (*BRM_home***/apps/pin_rel/SampleBatchControllerInfranet.properties**) to configure Batch Controller.

The default configuration for the sample Batch Controller runs RE Loader batch handler whenever a rated EDR file appears in the pipeline output directory. You can change this setting to trigger the RE Loader batch handler at specified times or based on other kinds of occurrences by editing the event entries that trigger the RE Loader batch handler.

To configure Batch Controller:

1. Copy the *BRM_home*/**apps/pin_rel/SampleBatchControllerInfranet.properties** file to your *BRM_home*/**apps/batch_controller** directory and change its name to **Infranet.properties**.
2. Open the *BRM_home*/**apps/batch_controller/Infranet.properties** file.
3. Edit the BRM connection parameters.
4. Set the **relHandler.start.string** parameter to the path of the RE Loader batch handler and to the name of the handler script that you gave it when configuring the RE Loader batch handler. See step 1 in "Configuring the RE Loader Batch Handler".

   For example:

   ```
   relHandler.start.string    BRM_home/apps/pin_rel/REL_handler_name.pl
   ```

5. (Optional) To change the number of RE Loader batch handler processes you want to run, set the maximum batch handler entries.

   > **Note:**
   >
   > The number of RE Loader batch handler processes that are called depends on the number of EDR files in the pipeline output directory. You should test your RE Loader performance if you change these entries. See "About Running Multiple RE Loader Processes" for more information.

   ```
   relHandler.max.at.highload.time 3
   relHandler.max.at.lowload.time 3
   ```

6. Set the **cdrFileEvent.file.location** parameter to specify the location of the pipeline output directory:

   ```
   cdrFileEvent.file.location /export/Portal/integRate
   ```

7. Set the **cdrFileEvent.file.pattern** parameter to which files should be processed by Batch Controller:

   ```
   cdrFileEvent.file.pattern cdr*.dat
   ```

   > **Note:**
   >
   > You can use an asterisk (*) as a wildcard character to represent zero or more characters in the file name.

8. Save and close the file.
9. Stop and restart Batch Controller.

Disabling Invoice Event Caching

If your system uses both RE Loader and invoicing, you must disable invoice event caching to ensure that invoices contain event details.

To disable invoice event caching:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Set the **event_cache** entry to **0**:

```
- fm_inv    event_cache    0
```

> **Note:**
>
> If this entry is set to any other value or is not present in the file, invoicing assumes there is data in the event cache and produces invoices without event details.

3. Save and close the file.
4. Stop and restart the CM.

Enabling a Billing Delay for CDRs

RE Loader cannot load a CDR for the next billing cycle when billing has not been completed unless delayed billing has been set. RE Loader requires delayed billing to be enabled so that it can attach events to the past bill cycle or current bill cycle according to the event time. If delayed billing is disabled, the CDRs for the next billing cycle are rated successfully, but RE Loader cannot load them.

You enable delayed billing by using the **ConfigBillingDelay** business parameter. See "Configuring Delayed Billing" in *BRM Configuring and Running Billing* for more information.

> **Note:**
>
> - If you use pipeline-triggered billing, you do not need to enable **ConfigBillingDelay** because RE Loader loads CDRs only for the current billing cycle.
> - If you do not use delayed billing, set **ConfigBillingDelay** to **0**.

Configuring Field Lengths for Input Data Files

Any value in the input data file that is longer than 255 characters must include its maximum size. If the maximum size is not specified, the value is truncated to 255 characters when it is loaded into the database.

Fields in the input data file that should not be loaded into the database are specified with the label FILLER in the SQL Loader control file. If the input data file contains a FILLER field with a value longer than 255 characters, SQL Loader will stop with an error indicating the field at fault. If this happens, add the maximum field size to the field entry in the SQL Loader control file. Use this syntax:

```
Field_name   FILLER CHAR(max_size)
```

For example:

```
DISCOUNT_INFO   FILLER CHAR(2000)
```

Configuring Whether to Perform Redo Generation

By default, RE Loader skips redo generation when loading files into the BRM database. This optimizes loading performance, but it can cause you to lose data if your system shuts down ungracefully.

You can re-enable redo generation by removing the UNRECOVERABLE option from each RE Loader control file.

To enable redo generation, do the following for each table's control file:

1. Open the *BRM_home*/**apps/pin_rel**/*control_file* file in a text editor, where *control_file* can be one of the files shown in Table 56-7.

Table 56-7 RE Loader Control Files

File Name	Description
event_bal_impacts_t.ctl	Control file for the EVENT_BAL_IMPACTS_T table.
event_delayed_act_wap_inter_t.ctl	Control file for the EVENT_DELAYED_ACT_WAP_INTER_T table.
event_delayed_session_gprs_t.ctl	Control file for the EVENT_DELAYED_SESSION_GPRS_T table.
event_sub_bals_t.ctl	Control file for the EVENT_SUB_BALS_T table.
event_sub_bal_imp_t.ctl	Control file for the EVENT_SUB_BAL_IMP_T table.
event_dlay_sess_tlcs_t.ctl	Control file for the EVENT_DLAY_SESS_TLCS_T table.
event_dlay_sess_tlcs_svc_cds_t.ctl	Control file for the EVENT_DLAY_SESS_TLCS_SVC_CDS_T table.
event_t.ctl	Control file for the EVENT_T table.
event_total_t.ctl	Control file for the EVENT_TOTAL_T table.
event_dlyd_session_tlco_gsm_t.ctl	Control file for the EVENT_DLYD_SESSION_TLCO_GSM_T table.

2. Comment out the UNRECOVERABLE option:

```
# UNRECOVERABLE
```

> **Note:**
>
> Removing the UNRECOVERABLE option significantly decreases loading performance.

3. Save and close the file.

57
Loading Prerated Events

This chapter describes how to run Oracle Communications Billing and Revenue Management (BRM) Rated Event (RE) Loader and provides information about troubleshooting and customizing.

For an overview on how RE Loader works, see "Understanding Rated Event Loader".

For information about configuring RE Loader, see "Configuring Rated Event Loader".

Loading Events Automatically

You can load events automatically in one of the following ways:

- By running RE Loader automatically by using Batch Controller and the RE Loader batch handler. When a prerated event file is available, Batch Controller automatically starts the RE Loader batch handler, which runs the RE Loader utility (**pin_rel**).

  To configure Batch Controller and the RE Loader batch handler, see "Configuring the RE Loader Batch Handler".

  > **Note:**
  >
  > Make sure you synchronize your rating and loading applications if you have configured the RE Loader batch handler to start the RE Loader utility to load rerated events.

- By running RE Loader daemon. See "Running the RE Loader Daemon".

Running the RE Loader Daemon

To run the RE Loader daemon:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor. *BRM_home* is the directory where you installed BRM components.
2. Specify how to run the RE Loader daemon by setting the entries given in Table 56-5.
3. Run the *BRM_home***/bin/start_rel_daemon** script.

   This script starts the RE Loader daemon.
4. Run the *BRM_home***/bin/stop_rel_daemon** script.

   This script stops the RE Loader daemon.

Loading Events Manually

> **Note:**
>
> Make sure you synchronize your rating and loading applications when running RE Loader.

To manually load pipeline-rated events, run the RE Loader utility (**pin_rel**) from the RE Loader directory.

Manually Loading Events from One Directory

If you have only one directory and need to load more than one event type, you must make sure **pin_rel** can find the prerated event data record (EDR) file. **pin_rel** looks for the EDR file in the directory specified in the **Infranet.rel.rated_event_file** entry in the **Infranet.properties** file. Before you run RE Loader manually, make sure the input EDR file is in this specified directory. To do this, do one of the following each time you run RE Loader:

- Move the EDR file to the directory specified in the **Infranet.properties** file.
- Change the **Infranet.rel.rated_event_file** entry in the **Infranet.properties** file to point to the directory containing the EDR file.

To run **pin_rel**, see "Running RE Loader Manually".

Manually Loading Events from Multiple Directories

If you have set up multiple directories, run **pin_rel** in the directory that corresponds to the service event type to load.

Running RE Loader Manually

You can manually run RE Loader from the command line in the following ways:

- **pin_rel** *event_file_name*

  This command loads events from *event_file_name* into the BRM database and then updates the account balances, bill items, and journals.

  Account balances, bill items, and journals are updated after all events have been loaded. If an error occurs during the loading phase, RE Loader cancels the process and all events loaded in the session are deleted from the BRM database.

  > **Note:**
  >
  > The name of the file in the command line can be found in the pipeline registry file. For more information, see "Configuring EDR Output Processing".

- **pin_rel** -**override** *event_file_name*

  This command starts an RE Loader process if one is not already running.

Only one RE Loader process can load the same database tables at the same time because each process locks the tables while loading them. When an RE Loader process is started, it checks the status of its last process and waits if the last process is not complete. However, if the process was manually canceled, the status may not indicate that the process has ended, even though it is no longer running. In this case, you use the **-override** option to start a new RE Loader process.

Monitoring and Maintaining RE Loader

To monitor and maintain RE Loader, perform the following:

- Troubleshooting Event Loading
- Preventing POID Errors in Multischema Systems
- Improving RE Loader Performance
- Customizing RE Loader
- Retrieving Data About Events You Load

Troubleshooting Event Loading

There are two distinct error-handling actions that RE Loader takes, depending on when the error occurs:

- If an error occurs while events are being loaded, the process is canceled and all events loaded in the session are deleted from the BRM database. The SQL loader errors are logged in a "bad" file (*BRM_home***/apps/pin_rel/***EDR_file_name***.bad**) and a fatal error is recorded in the RE Loader log file (*Processing_directory***/rel.pinlog**).
- If an error occurs while RE Loader is updating account balances, bill items, or journals, the loaded events are left in the database and an error is recorded in the RE Loader log file (*Processing_directory***/rel.pinlog**). If RE Loader stops due to errors while updating account balances, bill items, or journals, correct the problem and run RE Loader again.

Some error messages are sent to the console. To find out if an error occurred during rated event loading, check the **rel.pinlog** log file. See "Checking the RE Loader Log Files for Error Codes".

RE Loader checks for status in two places:

- The **/batch/rel** session status object.

  This object stores the status of the last RE Loader process. When you start RE Loader, it checks that status. If you try to reload a file that RE Loader has already successfully updated, the file is rejected because the session status indicates that the update for that file is complete.
- The REL_SUB_PROCESSES_T table.

  This tables stores information about loading errors that occurred during the preupdating stage. See "Checking for Errors that Occurred during the PreUpdate Process".

Checking the RE Loader Log Files for Error Codes

RE Loader uses the SQL Loader utility, **sqlldr**, to load events into the BRM database. The **sqlldr** process creates a new log file for each input file so that log files from a previous process are not overwritten.

The log files and the temporary files created during preprocessing incorporate the name of the input file in their file names, making it easier to debug if an error occurs.

Error codes follow the fully qualified error code (FQEC) scheme, which consists of a major code that represents the component and a minor code that represents the error number. All BRM-defined errors use a minor code from 0 through 99, and all custom errors use minor codes 100 and above.

For information on how to create custom error codes for RE Loader scripts and utilities, see "Creating Custom Error Codes".

> **Note:**
>
> Because modifying a stored procedure can corrupt data and cause maintenance and upgrade problems, custom error codes cannot be created for stored procedures.

The major and minor error codes for each RE Loader component are shown in Table 57-1.

Table 57-1 RE Loader Major and Minor Error Codes

Component	Description	Major Code	BRM Reserved Minor Codes	Customer Reserved Minor Codes
All	Universal code for success.	0	N/A	N/A
RE Loader driver	**pin_rel** script and Java driver code.	1000	0 - 999	N/A
Failure script	Script that is called when RE Loader attempts to load a data file that previously failed to load into the BRM database.	2000	0 - 99	100 - 255
Transform script	**pin_rel_transform_cdr.pl** script, which converts pipeline discount files into EDR format.	3000	0 - 99	100 - 255
Preprocess script	**pin_rel_preprocess_cdr.pl** script, which preprocesses the data files and creates bulk-loadable (**.blk**) files.	4000	0 - 99	100 - 255
Load utility	**sqlldr** utility, which loads data into the BRM database.	5000	0	1 - 999
Preupdate stored procedure	Stored procedure for updating the loaded data before releasing the partition to other RE Loader sessions.	7000	0 - 99	Not available
Update stored procedure	Stored procedure for updating account balances, bill items, and journals.	8000	0 - 99	Not available
Success script	Script that runs automatically when RE Loader successfully loads a data file into the BRM database.	9000	0 - 99	100 - 255
Database consistency check stored procedure	Stored procedure for verifying that the database indexes are correct before loading data into the database.	10000	0 - 99	Not available

Table 57-2 shows the BRM-defined error codes and messages, where *value* is the value returned in the error message:

Table 57-2 BRM-Defined Error Codes

RE Loader Error Number	Error Message
1000	REL encountered an error.
1002	The **infranet.rel.dbtype** properties value found is not supported: *value* Supported values are: *value*
1003	The **infranet.rel.partition_set_number** properties value found is not valid: *value* Valid values are between *value* and *value.*
1004	A table name properties value is missing for the given storable-class: *value*
1005	A duplicate table name properties value was found: *value*
1006	The **load_util** properties value is missing for the given storable-class: *value*
1007	A control file properties value is missing for the given storable-class: *value*
1008	The control file name could not be found in the command line.
1009	REL cannot be run until the Event Extraction Manager is complete.
1010	An unexpected SQL exception has occurred.
1011	An error occurred while attempting to connect to the BRM database.
1012	An error occurred while attempting to connect to the CM. Please validate the **infranet.connection** property value and ensure the CM is running.
1013	An error occurred while attempting to perform an opcode call.
1014	An interrupt has occurred and caused an error.
1015	The following file was not found: *value*
1016	An unexpected I/O error was encountered.
1017	The POID selected from the database sequence exceeds the maximum supported range of 2^{44}: *value*
1018	REL failed to select the partition name from the database.
1019	The poid_db could not be found in the input file.
1020	The poid_db found in the input file does not match the BRM database number for this CM connection. Found: *value* Expected: *value*
1021	The header record could not be found in the input file.
1022	The storable-class was not defined, or was not found in the header record.
1023	The time format found in the header record is not valid: *value*
1024	The creation process found in the header record is not supported: *value* Valid values are: *value*
1026	An invalid command-line was provided.
1027	The CM and JDBC BRM database connections are not configured to the same database schema.
1028	The REL session has timed out waiting for another REL session to complete.
1029	The file has previously completed successfully so it will not be loaded again: *value*
1030	The file is currently being processed by another REL session: *value*
1031	The *value* key is missing from the properties file.

Table 57-2 (Cont.) BRM-Defined Error Codes

RE Loader Error Number	Error Message
1032	The *value* value is missing from the properties file.
1033	The configured number of tables for this storable-class does not match the configured tables: *value*
1034	A number formatting error was encountered in the properties value for: *value*
1035	The **infranet.rel.updater_threads** properties value found is not valid: *value* Valid values are between *value* and *value.* To have REL auto-choose an appropriate number of threads, use the value: *value*
1036	An error occurred while attempting to parse a number for: *value*
1038	Cannot have control file with 'TRUNCATE' option when running REL in parallel loading mode between multiple REL processes.

Table 57-3 shows the BRM-defined failure script error codes.

Table 57-3 Failure Script Error Messages

Failure Script Error Number	Error Message
2000	The failure script encountered an error. The given command-line was: *value*
2001	The failure script command-line given arguments are not supported. The given command-line was: *value*
2002	The failure script command-line given flags value provided is not supported. The given command-line was: *value*
2003	The failure script command-line given directory could not be read. The given command-line was: *value*

Table 57-4 shows the BRM-defined transform script error codes.

Table 57-4 Transform Script Error Messages

Transform Script Error Number	Error Message
3000	The transform script encountered an error.
3001	The transform script command-line given arguments are not supported. The given command-line was: *value*
3002	The transform script command-line given input file could not be read. The given command-line was: *value*
3003	The transform script command-line given output file could not be created.
3004	The transform script command-line given negative discount carry over value is invalid. The given command-line was: *value*

Table 57-5 shows the BRM-defined preprocess script error codes.

Table 57-5 Preprocess Script Error Messages

Preprocess Script Error Number	Error Message
4000	The preprocess script encountered an error. The given command-line was: *value*
4001	The preprocess script command-line given arguments are not supported. The given command-line was: *value*
4002	The preprocess script failed to open a file.
4003	The preprocess script found the input file to be missing a balance record. The given command-line was: *value*
4004	The preprocess script found the input file to be missing a detail record. The given command-line was: *value*
4005	The preprocess script command-line given tables are not supported. The given command-line was: *value*
4006	The preprocess script command-line given increment_by value is not valid. The given command-line was: *value*
4007	The preprocess script did not find the expected number of records in the input file. The given command-line was: *value*
4008	The preprocess script found the input file to be missing an EDR record. The given command line was: *value* Used by SE Loader.
4009	The preprocess script did not find the expected EDR size for an EDR record. The given command line was: *value* Used by SE Loader.
4010	The preprocess script failed to parse fields mapping data for generating the control file. The given command line was: *value* Used by SE Loader.

Table 57-6 shows the BRM-defined load utility error codes.

Table 57-6 Load Utility Error Messages

Load Utility Error Number	Error Message
5000	The database load utility encountered an error.

Table 57-7 shows the BRM-defined insert stored procedure error codes.

Table 57-7 Insert Stored Procedure Error Messages

Insert Stored Procedure Error Number	Error Message
6000	The insert stored procedure encountered an error.

Table 57-8 shows the BRM-defined preupdate stored procedure error codes.

Table 57-8 Preupdate Stored Procedure Error Messages

Preupdate Stored Procedure Error Number	Error Message
7000	The preupdate stored procedure encountered an error.
7001	The preupdate stored procedure encountered an error on a select statement.
7002	The preupdate stored procedure encountered an error on an insert statement.
7003	The preupdate stored procedure encountered an error on an update statement.
7004	The preupdate stored procedure encountered an error on a delete statement.
7008	The preupdate stored procedure encountered a parsing error.
7010	The preupdate stored procedure could not find an item for an account.
7011	The preupdate stored procedure encountered an unexpected error.

Table 57-9 shows the BRM-defined update stored procedure error codes.

Table 57-9 Update Stored Procedure Error Messages

Update Stored Procedure Error Number	Error Message
8000	The update stored procedure encountered an error.
8001	The update stored procedure encountered an error on a select statement.
8002	The update stored procedure encountered an error on an insert statement.
8003	The update stored procedure encountered an error on an update statement.
8004	The update stored procedure encountered an error on a delete statement.
8008	The update stored procedure encountered a parsing error.
8009	The update stored procedure found its record is already being processed.
8010	The update stored procedure could not find an item for an account.
8011	The update stored procedure encountered an unexpected error.
8012	The update stored procedure encountered an invalid record count error.
8013	The update stored procedure encountered an error when updating the account balances.
8014	The update stored procedure encountered an error when updating the item balances.
8015	The update stored procedure encountered an error at TREL precommit.
8016	The update stored procedure encountered an error at TREL postcommit.

Table 57-10 shows the BRM-defined success script error codes.

Table 57-10 Success Script Error Messages

Success Script Error Number	Error Message
9000	The success script encountered an error. The given command-line was: *value*
9001	The success script command-line given arguments are not supported. The given command-line was: *value*

Table 57-10 (Cont.) Success Script Error Messages

Success Script Error Number	Error Message
9002	The success script command-line given flags value provided is not supported. The given command-line was: *value*
9003	The success script command-line given directory could not be read. The given command-line was: *value*

Table 57-11 shows the BRM-defined database consistency check error codes.

Table 57-11 Database Consistency Check Error Messages

Database Consistency Check Error Number	Error Message
10000	The database consistency check encountered an error.
10005	The database consistency check found an unpartitioned index.
10006	The database consistency check found an incorrectly partitioned index.
10007	The database consistency check found an unusable index.

Checking for Errors that Occurred during the PreUpdate Process

Errors that occur during the preupdate stage of the loading process are stored in the REL_SUB_PROCESSES_T table. To check for values in the table, run SQL*Plus.

Table 57-12 shows the error codes that are stored in the REL_SUB_PROCESSES_T table:

Table 57-12 Error Codes Stored in the REL_SUB_PROCESSES_T Table

Status Code	Status Number	Description
ERROR_SELECTING	–20001	An error occurred when selecting data from a table or tables.
ERROR_INSERTING	–20002	An error occurred during the insert process.
ERROR_UPDATING	–20003	An error occurred during the update process.
ERROR_DELETING	–20004	An error occurred during the delete process.
ERROR_UNPARTITIONED_INDEX	–20005	An error occurred because the index is not partitioned.
ERROR_INCORRECT_PART_INDEX	–20006	An error occurred because the index is global partitioned.
ERROR_UNUSABLE_INDEX	–20007	The index partitions are unusable.
ERROR_PARSING	–20008	An error occurred during the data parsing process.
ERROR_ALREADY_BEING_PROCESSED	–20009	An error occurred because the record is being processed by another thread.
ERROR_ITEM_NOT_IN_ACCOUNT	–20010	An error occurred because the item is already billed and **pre_updater_flag** is not enabled.
ERROR_UNEXPECTED	–20011	An unexpected error occurred in the pre-update procedure.
ERROR_UPDATE_ACCT_BALANCES	–20013	An error occurred while updating account balances.

Table 57-12 (Cont.) Error Codes Stored in the REL_SUB_PROCESSES_T Table

Status Code	Status Number	Description
ERROR_UPDATE_ITEM_BALANCES	–20014	An error occurred while updating item balances.

Fixing Event Loading Errors

In order to troubleshoot event loading errors, check the RE Loader log file *BRM_home***/apps/pin_rel/rel.pinlog**, where *BRM_home* is the directory in which you installed BRM components. See "Checking the RE Loader Log Files for Error Codes" for more information.

At times, when RE Loader fails, the **rel.pinlog** file does not list the error. If this occurs, check the status column in the BATCH_T table in the BRM database for the status of the REL process. Table 57-13 lists the status entries (and the corresponding code attributes).

Table 57-13 Status Entries in the BATCH_T Table

Status	Code Attribute
0	UPDATE_COMPLETE
1	LOAD_ERROR
2	UPDATE_ERROR
4	INSERT_ERROR
8	PREUPDATE_ERROR
16	REL_START
48	PRE_PROCESS
64	START_LOAD
80	LOADING
96	LOAD_COMPLETE
107	FAIL_TO_START_REL
240	PROCESS_LOADING
256	START_INSERT
512	INSERTING
768	INSERT_COMPLETE
1024	START_PREUPDATE
1280	PREUPDATING
1536	PREUPDATE_COMPLETE
3840	PROCESS_PREUPDATING
4096	START_UPDATE
8192	UPDATING
61440	PROCESS_UPDATING

The correct troubleshooting effort for an event loading error depends upon the error scenario:

- RE Loader fails to start:

  The RE Loader log file (**rel.pinlog**) displays the error code **107** (see Table 57-13)

In this error scenario, RE Loader failed to sart. The error occurs if REL is not running.

To troubleshoot this error, start REL using the following command:

```
rel<rated event file>
```

- Load failure:

  The RE Loader log file (**rel.pinlog**) displays the error code **5000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **1** (see Table 57-13).

  In this error scenario, RE Loader failed either before or during the loading of the events in the event file. The events are deleted from the event tables.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- RE Loader fails during the loading:

  The RE Loader log file (**rel.pinlog**) does not display any error. The status entry for the REL process in the BATCH_T table in the BRM database displays **80** (see Table 57-13).

  In this error scenario, RE Loader failed during the loading of the events and RE Loader was unable to update the session status or run the cleanup process.

  Use the **-override** option to start a new process to reload the events. For example:

  pin_rel -override *event_file_name*

  where *event_file_name* is the event file.

- Error occurs during the preupdate stored procedure:

  The RE Loader log file (**rel.pinlog**) displays preupdate stored procedure error codes starting at **7000** and below **8000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **8** (see Table 57-13).

  In this error scenario, RE Loader crashed during the execution of the preupdate stored procedure.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- RE Loader fails during the updating of events:

  The RE Loader log file (**rel.pinlog**) does not display any error. The status entry for the REL process in the BATCH_T table in the BRM database displays **8192** (see Table 57-13).

  In this error scenario, RE Loader crashed during the updating of the events and RE Loader was unable to update the session status or run the cleanup process.

  To troubleshoot this error, reload the events normally by using the same command to process the original event file.

- Error occurs during the update stored procedure:

  The RE Loader log file (**rel.pinlog**) displays update stored procedure error codes starting at **8000** and below **9000**. The status entry for the REL process in the BATCH_T table in the BRM database displays **2**.

  In this error scenario, RE Loader successfully loaded the events but failed during the execution of the update stored procedure. The BATCH_REL_SUB_PROCESSES_T table lists the last commit, indicating the point at which the database update failed.

  Reload the events normally. The update starts from this point.

Debugging Mismatches between Data Files and Control Files

RE Loader customizations can sometimes cause data files and control files to become unsynchronized, resulting in SQL Loader failures. To help you debug these situations, use the **pin_rel_enum_blk.pl** script, which enumerates fields in your bulk-loadable files. You can then manually compare the data file entries to the control file.

To debug mismatches between your data files and control files, enter the following commands:

```
% cd BRM_home/apps/pin_rel
% pin_rel_enum_blk.pl file_name [Line_num]
```

where:

- *file_name* specifies the name of the bulk-loadable file. For example, **test2.blk**.
- *Line_num* specifies the line number of the bulk-loadable file that you want to enumerate. The default is **1**.

Preventing POID Errors in Multischema Systems

BRM multischema systems ensure that all POIDs are unique across all database schemas by using a POID-generation algorithm. This BRM algorithm sets each schema's starting sequence number to a unique value and then increments each sequence number by a set value. By default, BRM sets the increment value equal to the number of schemas in your system.

For example, if your system contains three schemas:

- Schema 1 uses a starting sequence number of 10000
- Schema 2 uses a starting sequence number of 10001
- Schema 3 uses a starting sequence number of 10002

The incremental value is 3.

This example results in the following POID numbers shown in Table 57-14:

Table 57-14 Example Schema POID Numbers

Time	POID for Schema 1	POID for Schema 2	POID for Schema 3
1	10000	10001	10002
2	10003	10004	10005
3	10006	10007	10008

When RE Loader loads a batch of objects into the BRM database, it reserves a group of POIDs as follows:

1. Changes the increment value by using the following equation:

   ```
   (Number of objects to load) x (Current increment value)
   ```

   For example, if RE Loader needs to load 2,000 objects into the database and the current increment value is 3, it changes the increment value to 2,000 x 3 = 6,000.
2. Allocates POIDs to objects.
3. Returns the increment value to its original value.

However, if a major error occurs during the allocation process, the increment value can remain at the incorrect high value. To catch these situations, you can configure RE Loader to check the database increment value against a specified maximum before it reserves a group of POIDs. When the increment value exceeds the specified maximum, RE Loader exits and logs an error message, notifying your database administrator to manually reset the increment value.

To configure RE Loader to compare the increment value against a specified maximum:

1. Open the *BRM_home*/**apps/pin_rel/Infranet.properties** file in a text editor.
2. Set the **infranet.rel.max_increment_by** entry to the number of database schemas in your system:

   ```
   infranet.rel.max_increment_by = 20
   ```

   The default is **20**.
3. Save and close the file.

Improving RE Loader Performance

You can improve your RE Loader system performance by:

- Increasing the Number of Account Balance and Bill Item Updates
- Turning Off Index Verification to Improve Database Loading Performance
- Turning Off Database Verification to Improve Processing Performance
- Pruning Your RE Loader Control and Audit Tables

Increasing the Number of Account Balance and Bill Item Updates

RE Loader performance might be improved by increasing the number of account balance, bill item, and journal updates performed before committing the transaction.

You can modify the preupdate batch size and update batch size in the **Infranet.properties** file to specify how many updates to perform before committing the transaction. For example, if **updater_batch_size** is set to **5**, the stored procedure commits the transaction after every five updates. Increasing the number of updates might increase performance, but the updated account balances, bill items, and journals are not available until the transaction is committed. The default **batch_size** value is **5**.

> **Note:**
>
> Setting the **batch_size** value too high can result in deadlock. The value for best performance depends on your system configuration. You should test to find the best value for your system.

To change the **preupdater_batch_size** and **updater_batch_size** values:

1. Open the *BRM_home*/**apps/pin_rel/Infranet.properties** file in a text editor.

> **Note:**
>
> If you have already set up your RE Loader processing directories, make sure you edit the **Infranet.properties** file in each directory.

2. If necessary, edit the **infranet.connection** entry to point to the correct database.

   For example:

   ```
   infranet.connection=pcp://root.0.0.0.1:password@localhost:37180/service/pcm_client
   ```

3. Specify the preupdater batch size value in the **preupdater_batch_size** entry.

   For example:

   ```
   infranet.rel.default.preupdater_batch_size = 8
   ```

4. Specify the updater batch size value in the **updater_batch_size** entry:

   ```
   infranet.rel.default.updater_batch_size = 8
   ```

5. Save and close the file.

Turning Off Index Verification to Improve Database Loading Performance

By default, RE Loader automatically verifies that your indexes are correct before loading data into the BRM database. This extra step helps you discover configuration errors when testing your system in a development environment.

In production systems, however, you should turn off index verification to improve database loading performance.

When configured to verify indexes, RE Loader performs the following before it runs the SQL Loader utility:

1. Checks whether the indexes to load are partitioned, local, and usable.
2. Performs one of the following:
   - If the indexes are incorrect, RE Loader cancels the loading process and logs which indexes encountered problems.
   - If the indexes are correct, RE Loader runs the SQL Loader utility to load events into the database.

When configured to skip verification, RE Loader automatically runs the SQL Loader utility to load events into the database. When the indexes are incorrect, SQL Loader fails and RE Loader logs only that the database load utility encountered an error.

To turn off index verification:

1. Open the *BRM_home***/apps/pin_rel/Infranet.properties** file in a text editor.
2. Set the **Infranet.rel.validate_indexes** entry to **False**:

   ```
   Infranet.rel.validate_indexes = False
   ```

3. Save and close the file.

Turning Off Database Verification to Improve Processing Performance

By default, RE Loader automatically verifies that it is loading events into the correct database schema by validating the database number in the EDR file's first account object with the PCM

database number. This extra step helps you discover configuration errors when testing your multischema system in a development environment.

In production systems, however, you should turn off database verification to improve RE Loader database loading performance.

To turn off database verification:

1. Open the *BRM_home*/**apps/pin_rel/Infranet.properties** file in a text editor.
2. Set the **infranet.rel.validate_dbnumber** entry to **False**:

   ```
   infranet.rel.validate_dbnumber = False
   ```

3. Save and close the file.

Pruning Your RE Loader Control and Audit Tables

RE Loader control and audit tables grow indefinitely, so you should prune them periodically to increase system performance and reduce memory usage. To make pruning easier, you can use the RE Loader **purge_batch_rel_objects** stored procedure, which automatically prunes the tables for you.

To prune your control and audit tables:

1. Connect to the Oracle database with SQL*Plus:

   ```
   sqlplus system@DatabaseAlias
   Enter password: password
   ```

2. Enter the following command to run the stored procedure:

   ```
   SQL> pin_rel.purge_batch_rel_objects(int:Number)
   ```

   where *Number* specifies how many days worth of data to keep in the tables.

3. Type **exit** to exit SQL*Plus.

Customizing RE Loader

Some of the steps required to customize RE Loader should be performed by a programmer and database administrator. T

You can customize RE Loader by:

- Adding New Event Types for RE Loader to Load
- Creating Custom Error Codes

> **Note:**
>
> *Do not modify the **rel_updater_sp.sql** stored procedure or any other stored procedure.* Modifying a stored procedure can corrupt data and cause maintenance and upgrade problems. Stored procedures are delivered in source code format due to database limitations and are not designed to be modified. If you need to modify a stored procedure, you must obtain specific permission to do so from Oracle.

> **Note:**
>
> Sub-balances are stored in events in an internal format that optimizes performance and storage efficiency. As a result, the table that stores sub-balances is not visible in the data dictionary. You should not base your customizations on this specific internal format. All sub-balance data is accessible by using the BRM API. If you need to access this internal format, contact Oracle.

Adding New Event Types for RE Loader to Load

When you offer a new service, you create a new storable class for the service event type.

To use RE Loader to load events from a new service or new service subclass, you must create a delayed event type for your new service and configure RE Loader to load it.

It is possible to load a subclass of a preconfigured service event type without configuring that subclass. However, BRM will not be aware of the subclass because the subclass events will be inserted into the parent class table. To track the activity of the subclass events, you configure RE Loader to load the specific subclass.

You must create a new delayed event type for RE Loader prerated events. The new event storable class type must start with **/event/delayed**. For example, **/event/delayed/session/***new_event_type*.

> **Note:**
>
> Avoid loading prerated events by using RE Loader and another application such as an optional component or Universal Event Loader.

To add an event type for RE Loader to load:

1. If necessary, add the new event type storable class to BRM.

   > **Note:**
   >
   > If you installed GSM Manager, the **/telephony**, **/fax**, **/data**, and **/sms** subclasses of **/event/delayed/session/telco/gsm** already exist in the BRM database and do not need to be created. However, if you want to track activity specific to one of these subclasses, you must perform this entire procedure.

2. Create partitions for the event type by running the **partition_utils** utility from the *BRM_home***/apps/partition_utils** directory.

   For example, the following command creates partitions for **/event/delayed/session/telco/gsm** delayed events:

   ```
   partition_utils -o enable -t delayed -c /event/session/telco/gsm
   ```

> **Note:**
>
> You must create partitions for all subclasses of a specific service event type that you want to load.

3. Create a new control file for the new event type. A control file and format file specifies the format for a single database table (array or substruct). If you added new fields to an existing array or substruct, modify the control or format file for that table. If you added a new array or substruct, create a new control or format file for the new table. For instructions on creating a control or format file, see your Oracle documentation.
4. If you created or modified any control files, modify the RE Loader preprocess script (*BRM_home*/**apps/pin_rel/pin_rel_preprocess_cdr.pl**) to read the new event fields from the EDR data and write the fields to the files that are loaded by SQL Loader. You can follow the steps used for **/event/session/telco/gsm** in the **pin_rel_preprocess_cdr.pl** file as a guide.
5. Create a new RE Loader directory corresponding to the pipeline output directory. See "Setting Up RE Loader Processing Directories".
6. Add the following entries to the RE Loader **Infranet.properties** file in each directory:
   - The new event type
   - A new service record type corresponding to the new event type
   - The new control file that loads the new event
   - The new event tables that hold the new event type

   For information, see "Configuring the RE Loader Infranet.properties File" and the **Infranet.properties** file.
7. If you are running RE Loader automatically, you must also do the following:
   a. Configure the RE Loader batch handler in the new directory to load the new event type. See "Configuring the RE Loader Batch Handler".
   b. Add entries for the new RE Loader batch handler in the Batch Controller configuration file.

Creating Custom Error Codes

You can create custom error codes for RE Loader scripts and utilities by using the RE Loader **CustomErrorCodes.properties** file. You use this file to list your custom error codes and messages. All entries should follow the FQEC scheme and be grouped with the correct component. For more information, see "Checking the RE Loader Log Files for Error Codes".

To create custom error codes:

1. Modify the RE Loader script or utility to report the error. For more information, see the comments in the appropriate script or utility.
2. Open the *BRM_home*/**apps/pin_rel/CustomErrorCodes.properties** file in a text editor.
3. Add your custom error code to the file, making sure you use a minor code in the customer-reserved range.

   For example, the following entry creates a custom error code for the load utility:

   ```
   5100 = Sample load utility error message for a custom return code of 100.
   ```
4. Save and close the file.

Retrieving Data About Events You Load

BRM stores information about events loaded by RE Loader in a **/batch/rel** object. This object contains the input file name, number of records loaded, and other session information. To display the event object, use Billing Care.

> **Note:**
>
> If you use multiple database schemas, the **/batch/rel** object is created in the schema specified in the RE Loader **Infranet.properties** file.

Part VII
Setting Up Rerating

This part describes how to configure rerating, which is the process used to rerate events originally rated in batch by Pipeline Manager or in real time. This part covers the different tools used in rerating and the situations in which you use them. Part VII contains the following chapters:

58

About Rerating Events

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) rerating.

About Rerating

BRM rerating is the process you use to rerate events that were originally rated in batch by Pipeline Manager or in real time. You might rerate events for several reasons:

- To rerate a group of accounts after changing a charge offer
- To rerate an account when a customer service representative (CSR) backdates a subscriber's purchase or cancellation
- To back out events when call detail records (CDRs) contain field errors
- To rerate events when the rating conditions change during the session

> **Note:**
>
> – Member, child, and subordinate accounts in charge sharing, discount sharing, and account sponsorship and hierarchy groups are not automatically rerated when the parent or sponsoring account is rerated. For more information, see "About Rerating Events for Account Sharing Groups ".
>
> – You can rerate remittance accounts, but rerating subscriber accounts does not automatically rerate the remittance accounts. Events already included in remittance processing are not rerated.

The balance impact of rerating is the difference between the original rated event and the rerated event. This difference is applied to the account balance in the current billing cycle. For more information, see "How BRM Applies the Balance Impacts of Rerating".

About the Rerating Process

The rerating process consists of the following steps:

1. **Extracting events**. BRM extracts the events to rerate for an account or a group of accounts from the BRM database.
2. **Restoring balances**. BRM backs out the original event balance impacts and restores the account's discount and aggregation balances to their original states before rating the events.
3. **Calculating new balances**. BRM rerates the events based on new pricing information and calculates the new discount and aggregation balances.
4. **Recording the event**. BRM loads the events with the new balance impacts into the BRM database, updating all relevant account data.

About Automatic Allocation from Rerating

By default, BRM creates unallocated items for any adjustment items that are created as a result of rerating.

Corrective Billing and Automatic Allocation of Rerating Adjustments

If your corrective bills must contain the aggregation and allocation of automatic adjustments to the items on the original bill, you must enable the **AllocateReratingAdjustments** business parameter before you rerate the original bills. When the **AllocateReratingAdjustments** business parameter is enabled, adjustment details by original item by original bill are reported on the next bill for regular billing also.

To ensure that the corrective billing process includes or excludes these adjustments, check the setting for the **AllocateReratingAdjustments** business parameter *before* you run the rerating process.

If BRM has rerated a **/bill** object without allocating automatic adjustments (from the rerating) to the original bills, the corrective bill for that **/bill** object will not include the item adjustments generated by that rerating.

> **Note:**
>
> For such a **/bill** object, if you set the **AllocateReratingAdjustments** business parameter to **enabled** and rerun the rerating process for the bill, BRM does not allocate the adjustments.
>
> You must manually allocate the rerated items *before* you generate the corrective bill for such a **/bill** object.

If you enable BRM to automatically allocate the item adjustments generated by the rerating process to the original bill, BRM allocates adjustments to the bill items being corrected in the following manner:

- For open item accounting, the adjustment items are allocated to each bill that was corrected. Allocation is made to each of the original bills that included events or items that were corrected by these adjustments.
- For balance forward accounting, corrections are posted to the last bill only. Therefore, the rerating allocation is made to the final bill only. This final bill carries over the balances from all prior bill periods.

Enabling Automatic Allocation of Rerating Adjustments

To enable automatic allocation of rerating adjustments:

1. Go to the *BRM_home***/sys/data/config** directory, where *BRM_home* is the directory in which BRM is installed.
2. Run the following command, which creates an editable XML file from the **rerate** instance of the **/config/business_params** object:

```
pin_bus_params -r BusParamsRerate bus_params_rerate.xml
```

This command creates the XML file named **bus_params_rerate.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.

3. Open the **bus_params_rerate.xml.out** file.
4. Search for the following line:

   ```
   <AllocateReratingAdjustments>disabled</AllocateReratingAdjustments>
   ```

5. Change **disabled** to **enabled**.
6. Save the file as **bus_params_rerate.xml**.
7. Go to the *BRM_home***/sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the appropriate **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_rerate.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_rerate.xml** resides.

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing **rerate** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM **rerate** configuration.

9. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
10. Stop and restart the Connection Manager (CM).
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About Deferred Taxes During Rerating

By default, BRM does not compute deferred tax during rerating.

Enabling Calculation of Deferred Taxes During Rerating

> **Note:**
>
> When deferred taxation is enabled for rerating:
>
> - You cannot use selective rerating.
> - You cannot rerate events created by Bill Now.
> - You cannot rerate events that use multiple tax suppliers.

By default, BRM calculates taxes on any deferred taxable amount in the rerated events during the subsequent bill run. The rerated tax appears on the invoice for the subsequent bill run. Calculating taxes during rerating using the **ApplyDeferredTaxDuringRerating** business parameter enables corrected invoices to show the corrected tax amount. When combined with

corrective invoicing, rerated events will occur and their tax amounts will appear on the invoice for the original billing period.

To enable calculation of deferred taxes during rerating:

1. Go to the *BRM_home*/**sys/data/config** directory.
2. Run the following command, which creates an editable XML file from the **rerate** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

   This command creates the XML file named **bus_params_rerate.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.
3. Open the **bus_params_rerate.xml.out** file.
4. Search for the following line:

   ```
   <ApplyDeferredTaxDuringRerating>disabled</ApplyDeferredTaxDuringRerating>
   ```
5. Change **disabled** to **enabled**.
6. Save the file as **bus_params_rerate.xml**.
7. Go to the *BRM_home*/**sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the appropriate **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_rerate.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_rerate.xml** resides.

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing **rerate** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM rerate configuration.

9. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
10. Stop and restart the CM.
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About Rerating Events by Using the Rates Applied when the Rating Conditions Change During the Session

By default, events are rerated in order of occurrence based on the event end time. Rerating the events using the **OfferEligibilitySelectionMode** business parameter enables to rerate the events in order of event start time when the rating conditions change during the session. For example, if during a call session, the subscriber adds the called number of that session to a Friends and Family list, BRM applies the Friends and Family discount for the session from the time the called number is added to the Friends and Family list. When the call is rerated, the

actual rate is applied from the start time of the call until the called number is added to the Friends and Family list, and the discount is applied for the remaining session.

Enabling Rerating when the Rating Conditions Change During the Session

To enable rerating when the rating conditions change during the session:

1. Go to the *BRM_home*/**sys/data/config** directory, where *BRM_home* is the directory in which BRM is installed.
2. Run the following command, which creates an editable XML file from the **rerate** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

   This command creates the XML file named **bus_params_rerate.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.
3. Open the **bus_params_rerate.xml.out** file.
4. Search for the following line:

   ```
   <OfferEligibilitySelectionMode>endtime</OfferEligibilitySelectionMode>
   ```

5. Change **endtime** to **timeperiod**.
6. Save the file as **bus_params_rerate.xml**.
7. Go to the *BRM_home*/**sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the appropriate **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_rerate.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_rerate.xml** resides.

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing **rerate** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM **rerate** configuration.

9. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
10. Stop and restart the CM.
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Understanding the BRM Rerating Features

Using BRM, you can rerate events in the following ways:

- **By using Pipeline Manager to rerate pipeline-rated events only**. This method is used for rerating events originally rated by Pipeline Manager.

With this method, you extract the events for rerating from the BRM database by using the **pin_event_extract** utility. You then rerate the events in a batch rerating pipeline.

Events are rerated and recorded using a two-step process: Events are rerated by the pipeline and then recorded in the BRM database when they are loaded by Rated Event (RE) Loader. As a result, pipeline event balance impacts are applied when they are recorded in the BRM database (event creation time).

Because an account can also have real-time event balance impacts that are recorded in real time (event end time), there is a possibility that the pipeline-rated and real-time rated event balance impacts are applied out of sequence, which might affect how BRM bills for the usage.

> **Note:**
>
> When using Pipeline Manager to rerate pipeline-rated events, you must determine whether this method of rerating is sufficient to provide consistent results. Accounts with pipeline-rated usage may also have real-time events, such as purchase events, cancel events, or cycle fee events that can impact the rating of usage events.

- **By using the pin_rerate utility to rerate pipeline-rated and real-time rated events**. This method can be used for rerating events originally rated in real time and events originally rated in batch by Pipeline Manager.

  With this method, you select events for rerating and rerate the events by running the **pin_rerate** utility. Pipeline-rated events are rerated in a real-time rerating pipeline and real-time rated events are rerated by real-time rating opcodes.

  Pipeline-rated and real-time rated events are rerated and recorded in the same process. Events are rerated in order, based on the time they occurred (the event end time). This results in pipeline-rated *and* real-time-rated event balance impacts being applied in the correct sequence. For more information, see "About Rerating Pipeline and Real-Time Events Concurrently".

  If you do not use Pipeline Manager for batch rating, you can also use **pin_rerate** to rerate only real-time-rated events.

  > **Note:**
  >
  > Using **pin_rerate** to rerate events can degrade system performance, depending on the system load.
  >
  > When you have a high volume of events to rerate (for example, more than 100,000), you might be able to use Pipeline Manager to rerate pipeline-rated events and use **pin_rerate** to rerate real-time rated events to reduce the system load. You can do this if the accounts being rerated have only real-time-rated events or have only pipeline-rated events. If accounts have both real-time rated and pipeline-rated events, you should use **pin_rerate** to ensure consistent results.

About Rerating Pipeline and Real-Time Events Concurrently

When you use **pin_rerate** to rerate pipeline-rated and real-time-rated events concurrently, by default, events are rerated in order of occurrence based on the event end time and recorded in the BRM database as they are processed. This process applies the pipeline and real-time event balance impacts in the sequence that the original real-time usage occurred.

For example, R_1 and R_2 are real-time-rated events and P_1 and P_2 are pipeline-rated events. The sequence in which these events were generated in real time was R_1, P_1, P_2, R_2. However, the order in which they were recorded in the BRM database was R_1, R_2, P_1, P_2. When these events are rerated by **pin_rerate**, the events are rerated and recorded in the original sequence in which they occurred (R_1, P_1, P_2, R_2).

You can also use **pin_rerate** parameters to rerate the events in the order they were originally *recorded* (R_1, R_2, P_1, P_2).

What You Can Do with Rerating

When using Pipeline Manager to rerate pipeline-rated events only, you can do the following:

- Select accounts and events for rerating based on various criteria, such as:
  - Account number
  - Accounts that have events related to specific bundles, charge offers, discount offers, or specific event and service types
  - Events associated with specific charge offers, discounts, subscription services, or bill units
- Perform back-out-only rerating, which backs out the balance impacts of rating without reapplying new balance impacts.

When using **pin_rerate** to rerate pipeline-rated and real-time-rated events, you can do the following:

- Select accounts and events for rerating based on various criteria such as account number, bundles, charge offers, discounts, event type, service type, bill unit and so on.

  Rerating a bill unit rerates all usage events, cycle events, billing-time discounts, folds, and rollover events associated with the bill unit.
- Perform back-out-only rerating, which backs out the balance impacts of rating without reapplying new balance impacts.
- Rerate real-time-rated and pipeline-batch-rated events concurrently or, if you do not use Pipeline Manager for batch rating, rerate only real-time-rated events.
- Assign a rerate reason code to accounts selected for rerating. This enables you to rerate only accounts matching the reason for rerating.
- Set up automatic rerating, which automatically selects certain events for rerating so that you do not need to specify the accounts and events when you run the **pin_rerate** utility. You can set up the following automatic rerating features:
  - Automatic rerating for backdated events, rate changes, and rollover corrections
  - Out-of-order rerating for pipeline-rated events that are rated out of order (non-chronologically)

  - Trigger-dependent rerating for events based on custom rerating triggers that you configure. For example, you can automatically select events for rerating that are rated for a charge offer that has been canceled.
- Create custom **pin_rerate** parameters, which enables you to select events based on any event criteria.

About the Rerating Pipelines

Pipeline Manager rerates only the events that it previously rated. Pipeline Manager has two rerating pipelines. The one you use depends on whether you rerate events by using Pipeline Manager or by using **pin_rerate**:

- **The batch rerating pipeline**. This pipeline is used when rerating only pipeline-rated events by using Pipeline Manager. It receives and rerates events in batches. The rerated events are loaded into the BRM database in batches by RE Loader.
- **The real-time rerating pipeline**. This pipeline is used when rerating pipeline-rated events by using **pin_rerate**. It rerates events individually that it receives from the CM. The rerated events are recorded into the BRM database as they are processed.

How BRM Applies the Balance Impacts of Rerating

BRM rerating handles both billed and unbilled events and rerating events across billing cycles. When events include noncurrency balance impacts, BRM rerating properly redistributes noncurrency balance elements and charges or credits the account accordingly. Financial impacts to accounts receivable (A/R), general ledger (G/L), and taxation are taken into account when adjustments are made as a result of rerating.

When an event is rerated, BRM backs out the event from the BRM database by creating an adjustment event that fully negates the original balance impacts. This adjustment event can be either a shadow adjustment event or a regular adjustment event:

- If the event is unbilled, a shadow adjustment event is created. See "About Rerating Unbilled Events".
- If the event has already been billed or posted to the G/L, a regular adjustment event is created. See "About Rerating Billed and Posted Events".

For rerated usage events, the balance impacts of rerating are applied by the adjustment event. For rerated cycle events, a new cycle event is created that applies the balance impacts of rerating. Cycle events include cycle fees, folds, rollovers, and cycle discounts.

When the total rerating adjustment is zero, BRM does not generate a rerating adjustment event if the balance impacts of rerating and previous rating are equivalent. To determine if the balance impacts are equivalent, BRM compares the values of certain balance impact fields for rerating with previous rating, such as the balance element ID and balance group. If the values are different, BRM treats the rerated balance impacts as unique and generates rerating adjustment events.

You can customize how BRM determines whether the balance impacts of rerating and previous rating are equivalent by modifying the event balance impact fields that are used for comparison. For more information, see "Determining Whether Balance Impacts of Rerating and Previous Rating Are Equivalent".

How BRM Generates Rerated Events

BRM rerates events in chronological order:

- Usage events are rerated in order starting with the earliest usage event.
- For rollover, fold, and cycle discount events, BRM generates new events and applies the balance impacts according to the time that billing was run for the associated cycle.
- For cycle fee events, BRM generates new events and applies the balance impacts according to the time that the accounting cycle ends or when the charge offer is purchased or canceled.

Rollover, fold, cycle discount, and cycle fee events are generated in the order in which they logically occur: for example, billing-time discount events are generated before rollover events.

Rerating generates events in the following ways:

- By rerating the original event

  This process passes the event being rerated to the rating opcode. Rerating generates an adjustment event to apply the balance impacts of rerating. This is the most basic rerating process.

- By reapplying business logic based on information in the original event

  This process passes information from the event being rerated to the opcode that generated the event. The opcode may or may not generate a new event. This process is used when the event attributes may be different after rerating. For example, a charge offer's purchase fee may have been changed since the purchase fee event was generated. To reapply the purchase fee, rerating passes the charge offer information in the original bundle-purchase event to the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode, which generates a new purchase fee event, if applicable.

  > **Note:**
  >
  > In some cases, purchase and cancellation events are not available during rerating. In such cases, BRM rerates the purchase-fee or cancellation-fee event. For more information, see "Events That Are Not Rerated".

- By reapplying business logic independent of the original event

  This process performs business logic that may generate new events even if there is no existing event. With this process, information in the original event is not used. Instead, the business logic that generated the original event is reapplied and a new event is generated, if appropriate.

  For example, when rerating an account for a specific period, if the events selected for rerating are associated with charge offers that have a cycle fee, rerating calls the cycle fee opcode and generates a new cycle fee event based on the charge offer's current rates. This is done independent of any existing cycle fee event. If there is an existing cycle fee event, it is replaced by the corresponding new cycle fee event generated by rerating.

Table 58-1 shows the rerating process for specific event types.

Table 58-1 Rerating Process and Event Type

Event Type	Order of Rerating	How Rerated
Usage	Chronological, based on event time	Rerate the original event.

Table 58-1 (Cont.) Rerating Process and Event Type

Event Type	Order of Rerating	How Rerated
Cycle fee	Generated according to the time that the charge offer is purchased or canceled or when the associated accounting cycle starts or ends, and based on the logical order of events	Reapply business logic independent of the original event.
Rollover	Generated according to the time that the associated billing cycle ends, and based on the logical order of events	Reapply business logic independent of the original event.
Fold	Generated according to the time that the associated billing cycle ends, and based on the logical order of events	Reapply business logic independent of the original event.
Billing-time discount	Generated according to the time that the associated billing cycle ends, and based on the logical order of events	Reapply business logic independent of the original event.
Purchase	Chronological, based on event time	Reapply business logic based on information in the original event.
Cancellation	Chronological, based on event time	Reapply business logic based on information in the original event.

About Rerating Unbilled Events

When unbilled events are rerated, a shadow adjustment event is created. This event is called a *shadow event* because its balance impact is added to the original event's bill item rather than to an adjustment item.

Rerating Unbilled Usage Events

When unbilled usage events are rerated, the shadow adjustment event fully negates the original balance impacts and applies new balance impacts for the rerated amount.

Rerating Unbilled Cycle Events

When unbilled cycle events are rerated, the shadow adjustment event fully negates the original balance impacts. Then a new cycle event is created that applies the rerated balance impacts. The new cycle event is of the same event type as the original event.

> **Note:**
>
> When the purchase start time is the same as the current accounting cycle end date, the cycle forward fee balance impact is applied to the next month's bill instead of the current month's bill when billing is run after rerating is performed.
>
> For example, suppose an account is created on January 1 with a cycle forward fee of $20 and the purchase start time is deferred by 1 month (set to the accounting cycle end date). On February 1, when **pin_rerate**is run from January 1, because the purchase start time is the same as the accounting cycle end date, rerate internally triggers automatic billing. The billing process changes the status of the bill item to open. Because the bill item status is now open, rerating applies the rerate adjustments to the next bill. When regular billing is run on February 2, the cycle fee is not applied to the January bill; instead, it is applied to the February bill. For more information about rerating billed events, see "About Rerating Billed and Posted Events".

About Rerating Billed and Posted Events

When you rerate events that are already billed and unbilled events that are already posted to the G/L, the results of rerating are applied as adjustments to the next bill, which is the bill for the current cycle. The balance impacts are applied to the adjustment item.

Rerating Billed and Posted Usage Events

When usage events that are billed or posted to the G/L are rerated, an adjustment event fully negates the original balance impacts and applies new balance impacts for the rerated amount.

Rerating Billed and Posted Cycle Events

For billed cycle fee, fold, rollover, and cycle discount events, an adjustment event is created that fully negates the original balance impacts. The adjustment is applied to the current billing cycle and does not impact the original event's items. A new cycle event is then created that applies the rerated balance impacts. The new cycle event is of the same event type as the original event and is applied to the current cycle.

When billing is run, if a noncurrency balance element has a zero balance, no cycle event is generated for that balance element. For example, if there are no remaining free minutes to roll over at billing time, no rollover event is created. However, if rerating creates a nonzero amount for a balance element that was previously zero when billed, the rerating process generates the appropriate cycle event.

About Rerating and Pricing Changes

Changes to account charge offers are audited. If you change a charge offer attribute after events are rated, and then rerate those events, the rerating process uses only the current account subscription data.

Events That Are Not Rerated

Usually, purchase-fee events (**/event/billing/product/fee/purchase**) and cancellation-fee events (**/event/billing/product/fee/cancel**) are not directly rerated. Instead, information in the

original bundle purchase event (**/event/billing/deal/purchase**) and product cancellation event (**/event/billing/product/action/cancel**) is resent to the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL and PCM_OP_SUBCRIPTION_CANCEL_PRODUCT opcodes respectively. The opcodes generate new fee events, if applicable.

In some cases, the purchase-fee and cancellation-fee events are directly rerated because the purchase and cancellation events may not be available for rerating: for example, when rerating deferred purchase fees or when using selective rerating in which only purchase-fee or cancellation-fee events are specified.

The following events are not rerated but are reapplied because they were not originally generated by rating:

- **/event/billing/adjustment**
- **/event/billing/charge**
- **/event/billing/cycle/tax**
- **/event/billing/debit**
- **/event/billing/dispute**
- **/event/billing/item**
- **/event/billing/payment**
- **/event/billing/refund**
- **/event/billing/reversal**
- **/event/billing/settlement**
- **/event/billing/writeoff**

About Rerating Events for Account Sharing Groups

When rerating accounts and bill units in a hierarchy, sponsor, or discount sharing group, BRM does not automatically rerate subordinate and member accounts and bill units. To rerate subordinate and member accounts and bill units, you must specify them when you select the accounts for rerating. However, when BRM rerates an event for a service associated with sponsorship, rerating impacts the appropriate sponsoring or sponsored account's balance.

About Rerating Events That Impact Sponsored Balance Elements

If the balance impacts of an event are sponsored by another account, the balance impacts of rerating are applied to the sponsoring account. For example, account A pays for Internet access for account B. If account B's events are rerated, the balance impacts of rerating the Internet usage events are applied to account A's balance.

Likewise, if a rerated event impacts a balance element that is shared with another account, the balance element in the sponsored account is adjusted. For example, account A gets 1 frequent flyer mile for every dollar that account B spends. If account B's events are rerated and its current balance is reduced from $100 to $50, account A's frequent flyer miles are adjusted from 100 to 50.

If a rerated, sponsored event impacts a balance element that is shared by more than two accounts, rerating adjusts only the balances in the rerated account and the sponsoring account. For example, account A shares free minutes with accounts B and C. If account B and account C both make calls, and then account B is rerated, any impact on the free minutes

account B used is applied to account A. However, that free-minute balance impact in account A is not carried over to account C, even though account C shares those balance elements.

BRM Functionality Affected by Rerating

When you rerate events, the following areas in BRM are affected:

- General ledger entries. See "Determining the G/L Entry for an Event".
- Invoices. See "Displaying Shadow-Event Rerating Details on Invoices".

Determining the G/L Entry for an Event

The G/L entry is determined at the time of rating. If you rerate as a result of pricing changes, the G/L entry could change. Whether you record a shadow event or an adjustment event, you must determine the correct balance to be posted in the G/L.

When recording a shadow event, the G/L ID of the rerating balance impacts have the same G/L ID as the original event's balance impacts. If you changed the G/L ID after the original event was rated, the balance impacts of rerating use the new G/L ID.

When recording an adjustment, you can configure the G/L ID to use. For example, you can use the same G/L ID as the original event, a new G/L ID, or an adjustment G/L ID (a separate G/L ID bucket to record all adjustments as part of rerating). The adjustment G/L ID can be the same or different than the G/L ID used for regular (not rerated) event adjustments.

Displaying Shadow-Event Rerating Details on Invoices

When recording a shadow event, you can customize your invoice to either show the details of rerating or show the result of rerating only in an end balance.

To customize your invoice:

1. Open the *BRM_home***/sys/cm/pin.conf** file.
2. Set the **show_rerate_details** entry:
   - To show the details of rerating, set the entry to **1**.
   - To show the result of rerating only in an end balance, set the entry to **0.** The default is **0**.

   For example:

   ```
   - fm_inv_pol show_rerate_details 0
   ```
3. Save and close the file.
4. Stop and restart the CM.

Determining Whether Balance Impacts of Rerating and Previous Rating Are Equivalent

When usage events are rerated, an adjustment event is generated to apply the balance impacts of rerating. The total balance impacts of the adjustment event is the difference between the previous rating and rerating results.

When rerating produces the same result as previous rating, creating an adjustment event is not necessary. However, because an event can include multiple balance impacts, BRM must ensure that the balance impacts in each rating process are actually equivalent.

To determine whether the balance impacts of rerating and previous rating are equivalent, BRM checks the following values of the balance impacts in the event. If any of these values is different for rerating than it is for previous rating, BRM treats the associated balance impact as being unique and generates an adjustment event that includes the balance impact:

- The ID of the balance element being impacted
- The ID of the balance group being impacted
- The G/L ID
- The tax code
- The charge offer used to rate the event

The list of default balance impact fields used to compare the balance impacts of events' rerating and previous rating results is stored in the **/config/rerate_flds/compare_bi** object in the BRM database. You can modify the balance impact fields stored in the **/config/rerate_flds/compare_bi** object to customize how BRM determines whether balance impacts are equivalent.

For example, if you remove the charge offer field and the total rerating adjustment is zero, adjustment events are not generated even when the charge offer used to rerate events is different than the charge offer previously used (providing the other balance impact fields are the same). Or, if you specify additional balance impact fields, adjustment events are generated when the values of those fields differ. For example, you can specify the impact category field to record the rerating adjustment when calls are made from different locations even though rerating results in the same total charge.

To customize the list of event balance impact fields that determine whether rerated and previously rated balance impacts are equivalent, see "Specifying How to Compare Balance Impacts When Creating Adjustment Events".

Specifying How to Compare Balance Impacts When Creating Adjustment Events

BRM checks the **/config/rerate_flds/compare_bi** object during rerating. If the balance impact fields in this object have the same values in the event for both rerating and previous rating, and the total rerating adjustment is zero, the results of rerating and previous rating are considered equivalent and no adjustment event is created.

> **Note:**
>
> The **load_pin_rerate_flds** utility overwrites existing balance-impact comparison fields. If you are updating balance-impact comparison fields, you cannot load new fields only: you must load complete sets of the balance-impact comparison fields when you run the **load_pin_rerate_flds** utility.

> **Note:**
>
> The **load_pin_rerate_flds** utility uses a configuration file (**pin.conf**) located in the same directory to connect to the BRM database. Edit the configuration file to connect to your BRM database.

> **Note:**
>
> The **load_pin_rerate_flds** utility loads extraction keys that define custom **pin_rerate** parameters. You cannot use the same file to specify extraction keys and balance-impact comparison fields; you must run **load_pin_rerate_flds** with separate files for each configuration.

To customize the fields used to compare the event balance impacts of rerating with previous rating:

1. Open the *BRM_home***/sys/data/config/pin_rerate_compare_bi.xml** file.

   Add a **RerateCompareBalImpacts** element for each event balance impact field. The following balance impact fields are mandatory:

   - PIN_FLD_RESOURCE_ID
   - PIN_FLD_BAL_GRP_OBJ
   - PIN_FLD_GL_ID
   - PIN_FLD_TAX_CODE

   > **Note:**
   >
   > PIN_FLD_PRODUCT_OBJ is specified in the **pin_rerate_compare_bi.xml** file by default, but it is an optional field.

   You can specify any additional field from the **/event** object's PIN_FLD_BAL_IMPACTS array.
2. Save and close the file.
3. Run the following command, which loads the contents of the XML file into the **/config/rerate_flds/compare_bi** object:

   ```
   load_pin_rerate_flds pin_rerate_compare_bi.xml
   ```

   If you do not run the utility from the directory in which the XML file is located, you must include the complete path to the file. For example:

   ```
   load_pin_rerate_flds BRM_home/sys/data/config/pin_rerate_compare_bi.xml
   ```
4. Stop and restart the CM.
5. To verify that the balance-impact comparison fields were loaded, display the **/config/rerate_flds/compare_bi** object by using Object Browser, or use the **robj** command with the **testnap** utility.

How BRM Tracks Rerated Events

This section is useful if you write custom code that requires accessing events that were rerated.

To back out original rated events, BRM generates adjustment events (**/event/billing/adjustment/event**). These events contain a compliment of all balance impacts in the original events to fully negate the events. When an event is rerated, the PIN_FLD_RERATE_OBJ field in the original rated event references this backout adjustment event.

When usage events are rerated, the new balance impacts of rerating are included in the same adjustment event that backs out the original balance impacts. This means the original event references the new event that contains the rerated balance impacts.

When cycle fee, cycle discount, fold, and rollover events are rerated, new cycle events are created to apply the balance impacts of rerating. In this case, there is no relationship between the original events and the new (cycle) events that contains the rerated balance impacts.

Because you can use flexible cycles and multiple discounts, there might be more than one rollover, fold, and discount event to rerate for a given billing cycle:

- A fold event is generated per balance for each balance element that has a fold.
- A rollover event is generated per balance group for each charge offer that has a rollover.
- A discount event is generated for each cycle discount.

59

About Real-Time Rerating Pipelines

This chapter describes:

- How events are processed by Oracle Communications Billing and Revenue Management (BRM) real-time rerating pipelines.
- How to configure real-time rerating pipelines.

About Real-Time Rerating Pipelines

When you run the **pin_rerate** utility to rerate events rated in real-time and events rated in a batch pipeline, you run the **pin_rerate** utility to send pipeline-rated events to a real-time rerating pipeline to be rerated.

The Connection Manager (CM) sends pipeline-rated events in the form of an flist to the NET_EM pipeline module. The NET_EM module transfers the event to a real-time rerating pipeline for rerating.

The real-time rating opcodes add to the flist all the enrichment data needed by the real-time rerating pipeline to rate the event. For example, if the real-time rating opcodes determine that a discount should be applied, it adds the discount information to the flist.

Similar to a batch rerating pipeline, a real-time rerating pipeline performs both rating and discounting functions.

To rerate events, the real-time rerating pipeline gets the new pricing information from the Pipeline Manager database. Certain configuration data, such as currency and noncurrency balance element information, are obtained from the BRM database.

Overview of Event Processing in the Real-Time Rerating Pipeline

1. The INP_Realtime input module converts the flist received from the NET_EM module to event data record (EDR) format and creates an EDR container.

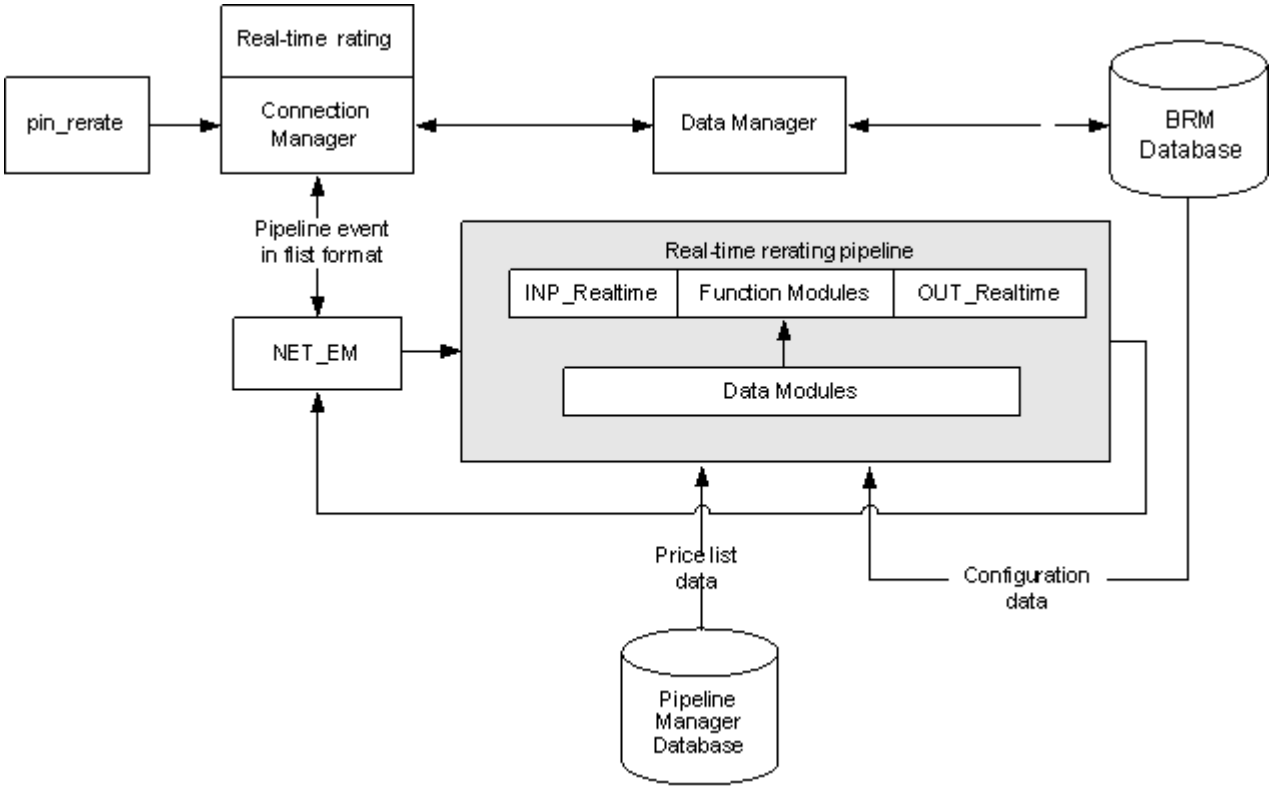
   > **Note:**
   >
   > You can also create and run an iScript to manipulate data in the EDR container before it is sent to the pipeline modules.

2. The pipeline function modules calculate the new balance impacts by using the new pricing and discounting information and then add the balance impacts to the EDR container.
3. The OUT_Realtime module converts the EDR back into flist format and sends the flist to NET_EM.
4. NET_EM sends the flist with the new balance impacts back to the CM.
5. BRM updates the account's balances and records the event in the BRM database.

For detailed information about configuring the real-time rerating pipeline, see "Configuring Rerating of Pipeline-Rated Events in the Real-Time Rerating Pipeline".

Figure 59-1 shows the data flow for rerating pipeline-rated events by using the real-time rerating pipeline:

Figure 59-1 Rerating Data Flow for Pipeline-Rated Events



About Transaction Management for the Real-Time Rerating Pipeline

The real-time rerating pipeline does not use the Transaction Manager (TAM) module. Instead, transaction handling is provided by the CM. When a rerating transaction fails, the CM rolls back the transaction and restores all the account balances in the BRM database. For this reason, function modules and iScripts used by the real-time rerating pipeline are stateless.

> **Note:**
>
> If you use iScripts for custom processing, ensure that the iScripts are stateless.

Configuring Rerating of Pipeline-Rated Events in the Real-Time Rerating Pipeline

To configure rerating of pipeline-rated events in the real-time rerating pipeline, perform the following tasks:

- Configure the real-time rerating pipeline. See "Configuring a Real-Time Rerating Pipeline".

> **Note:**
>
> When you configure pipelines in the **Pipelines** section, you must add the pipelines to the IFW_PIPELINE table. Otherwise, you will receive an error at system startup. For more information, see "Configuring the Real-Time Rerating Pipelines in the IFW_PIPELINE Table".

- Configure the CM to send rerate requests to the NET_EM module.
- Configure NET_EM to route rerate requests to the real-time rerating pipeline.

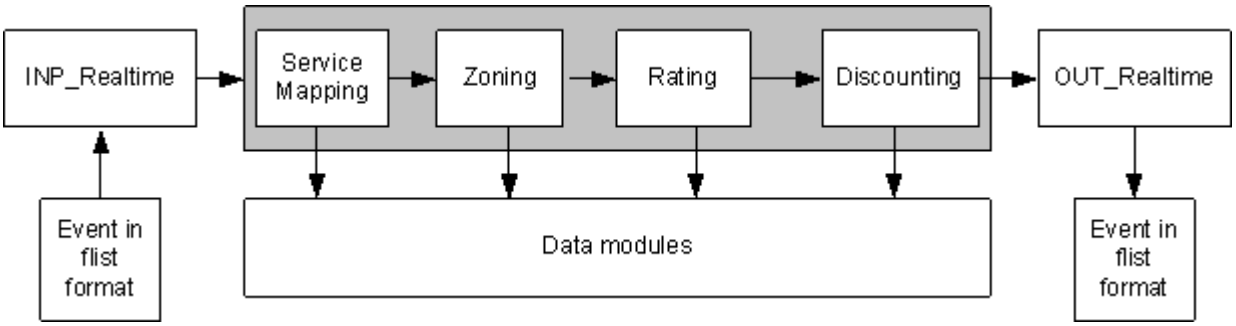
Configuring a Real-Time Rerating Pipeline

A real-time rerating pipeline is configured similarly to a batch rerating pipeline, but with fewer preprocessing and post-rating modules. This is because most of the enrichment data is provided in the input flist received from the CM.

You typically create a separate instance of Pipeline Manager for real-time rerating. The default registry file is *pipeline_home***/conf/wirelessRealtime.reg**.

Figure 59-2 shows the real-time rerating pipeline architecture:

Figure 59-2 Real-Time Rerating Pipeline Architecture



Configuring Multiple Real-Time Rerating Pipelines

To improve performance or scalability, you configure multiple instances of real-time rerating pipelines. For example, you can increase performance by configuring multiple pipelines to process rerate requests in parallel or by configuring multiple real-time rerating pipelines to process different rerate requests: for example, by configuring one pipeline that rerates only GPRS events and another that rerates only GSM events.

Configuring the Real-Time Rerating Data Pool

Configure the following data modules:

- DAT_Zone
- Rateplan

- DAT_Calendar
- DAT_TimeModel
- DAT_PriceModel
- Dayrate
- DAT_Currency
- DAT_USC_Map

Configuring the Modules in the Real-Time Rerating Pipeline

To configure the modules in the real-time rerating pipelines, perform the following tasks:

- Configure the input module to use the XML file containing the flist-to-EDR mappings.

  Use this entry for the **OpcodeMapping** entry:

  ```
  OpcodeMapping = ./formatDesc/Formats/Realtime/rate_event.xml
  ```

- Configure NET_EM to manage data between the CM and Pipeline Manager.
- Configure the output module to add any customizations to the output flist.
- Configure these function modules to perform rerating:
  - FCT_ServiceCodeMap
  - FCT_CustomerRating
  - FCT_PreRating
  - FCT_IRules
  - FCT_USC_Map
  - FCT_RSC_Map
  - FCT_MainRating
  - FCT_Dayrate
  - FCT_RateAdjust
  - FCT_Rounding
  - FCT_DiscountAnalysis
  - FCT_Discount

Configuring the Real-Time Rerating Pipeline to Set Charge Offer Validity Periods

If your charge offers are configured to start when they are first used, configure the ISC_FirstProductRealtime iScript in the real-time rerating pipeline.

When charge offers start on first usage, the real-time rerating pipeline adds the account's first-usage charge offer and discount information to the EDR. ISC_FirstProductRealtime sets the validity period in the BRM database for those charge offers that were used to rate the event and that start on first usage. This triggers any purchase and cycle fees.

Configuring the Real-Time Rerating Pipelines in the IFW_PIPELINE Table

Pipeline Manager stores information about pipelines in the IFW_PIPELINE table. The pipelines that are preconfigured in the **Pipelines** section of the default registry file (*pipeline_home***/conf/**

wirelessRealtime.reg) are inserted into the IFW_PIPELINE table during Pipeline Manager installation.

> **Note:**
>
> - If you are *not* using the default registry file, and you have configured new real-time rerating pipelines, you must manually insert the pipelines into the IFW_PIPELINE table by using SQL commands. Otherwise, you will receive an error at system startup.
> - If you *are* using the default registry file and have changed the default pipeline names or you have configured additional pipelines, you must manually insert the new pipelines into the IFW_PIPELINE table.

The following example shows SQL commands to insert RealtimeReratingGSM and RealtimeReratingGPRS pipelines into the IFW_PIPELINE table:

```
% sqlplus pin/password@databaseAlias

SQL>INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC ) VALUES
( 'RealtimeReratingGSM', 'GSM Realtime Rerating Pipeline', 'ALL_RATE');

SQL>INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC ) VALUES
( 'RealtimeReratingGPRS', 'GPRS Realtime Rerating Pipeline', 'ALL_RATE');

SQL>commit;
```

Configuring NET_EM to Route Rerate Requests Based on the Event Field Value

To configure NET_EM to route rerate requests to multiple real-time rerating pipelines based on the type of event, you set the **FieldName** and **FieldValue** NET_EM module registry entries.

> **Note:**
>
> If you use event routing based on the event field value, ensure that the input events contain the expected field name and field values specified in the NET_EM module registry. Otherwise, NET_EM will not be able to route the events.

By using the "**.**" notation, you can specify a field at any level in the input event flist to be used to route the event. For example, this substruct and field:

```
PIN_FLD_EVENT
     PIN_FLD_POID
```

is represented like this:

```
PIN_FLD_EVENT.PIN_FLD_POID
```

In the NET_EM registry below, if PIN_FLD_EVENT.PIN_FLD_POID is a GSM event, the rerate request is routed to any one of the two instances of the GSM rerating pipeline (RealtimeReratingGSM). If the event is a GPRS event, the rerate request is routed to any one of the two instances of the GPRS rerating pipeline (RealtimeReratingGPRS).

```
DataPool
{
     RealtimePipeline
     {
          ModuleName = NET_EM
          Module
          {
               ThreadPool
               {
                   Port = 14579
                   Threads = 4
               }

               ReratingOpcode
               {
                   OpcodeName = PCM_OP_RATE_PIPELINE_EVENT
                   FieldName = PIN_FLD_EVENT.PIN_FLD_POID
                   GSMEvent
                   {
                     FieldValue = /event/delayed/session/telco/gsm
                     PipelineName = RealtimeReratingGSM
                     NumberOfRTPipelines = 2
                   }
                   GPRSEvent
                   {
                     FieldValue = /event/delayed/session/gprs
                     PipelineName = RealtimeReratingGPRS
                     NumberOfRTPipelines = 2
                   }
               }

          }
     }
}
```

60

About Rerating Pipeline-Rated Events

This chapter provides an overview of how you can rerate events that were originally rated in batch by Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Overview of Rerating Events

It is possible to discover pricing or rating configuration errors after events have been rated. For example, a sample review of invoices might reveal a price configuration error. In addition to correcting the price configuration, you might need to rerate events that were rated with the incorrect configuration.

Use the following components to rerate events that were originally rated by Pipeline Manager:

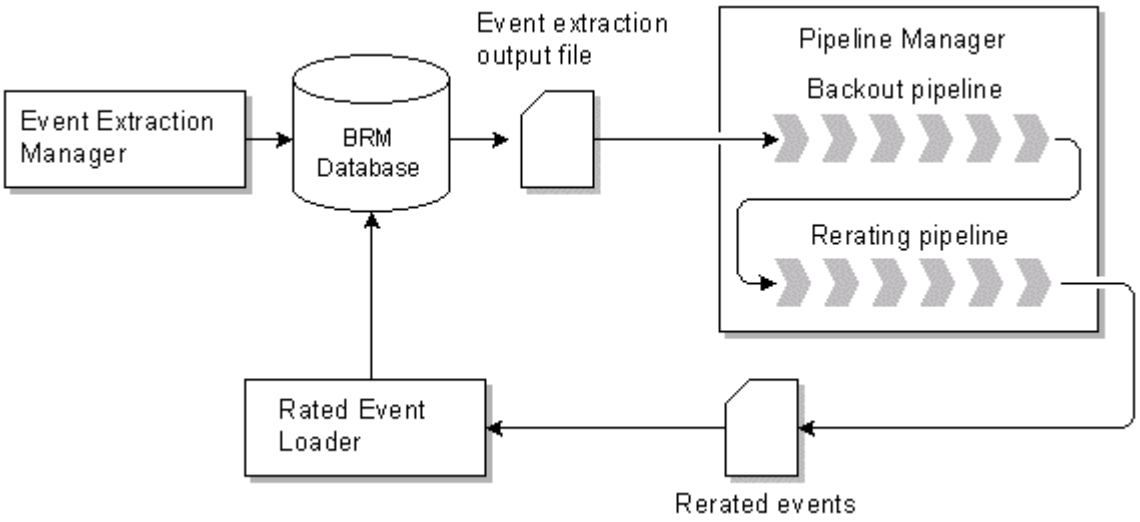
1. Use the *Event Extraction utility* to obtain data from the BRM database about the events you must rerate. This utility writes event data to a file.
2. Use *Pipeline Manager* to process the file that was created by the Event Extraction utility. There are two steps:
   a. Restore discount and aggregation data to the state it was in before rating the events. For example, if you use volume discounting based on calls made, you can adjust the number of calls made.
   b. Rerate the events using the new rating data. This process creates an output file.
3. Use the *Rated Event Loader (RE Loader)* to load the output file generated by Pipeline Manager into the BRM database.

Pipeline Manager only rerates events that it has previously processed. When you use pipeline rerating, there might be events that are not rerated. For example, if you retrieve events for rerating based on the account, any events belonging to the account that were generated in BRM and not previously processed by Pipeline Manager are not rerated. Ensure that all events are rerated by also running the BRM rerating utility (**pin_rerate**).

> **Note:**
>
> When you rerate wireless events by using Pipeline Manager, you rerate only EDRs that have been rated by Pipeline Manager. Events that have been rated in real time are rerated by using the **pin_rerate** utility.

Figure 60-1 shows the rerating process:

Figure 60-1 Rerating Process



About Extracting Events from the BRM Database

The Event Extraction Manager searches for and extracts events meeting a specified search criteria and writes the results to an output file. It does not modify or process any of the events, nor does it lock accounts in your database. This enables BRM and other applications to safely access accounts in the database while you are extracting events.

To extract events, you perform the following steps:

1. Edit the event search file to specify which events to extract. You can specify events based on many different event attributes, for example, date, account, charge offer, charge, and service.
2. Configure and run the **pin_event_extract** utility to create the event extract output file. This file contains the list of events to rerate in EDR (event data record) format, and is used as the input for the Pipeline Manager rerating process.

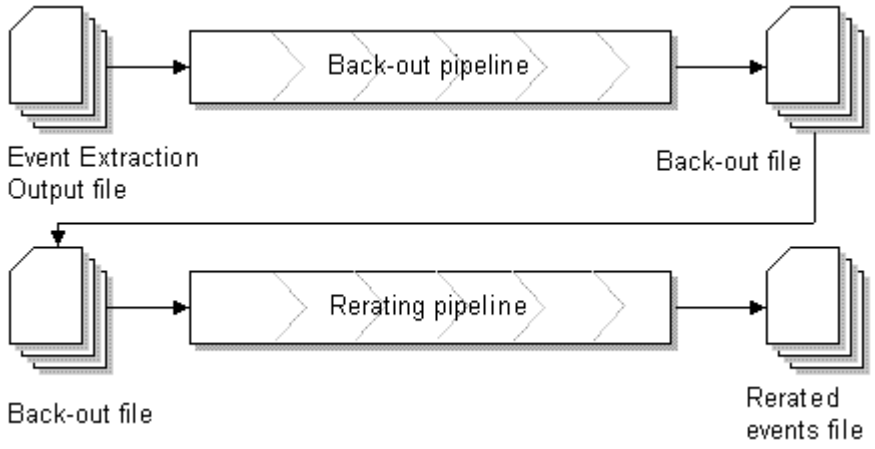
About Rerating Events with Pipeline Manager

To rerate events with Pipeline Manager, you perform the following steps:

1. Correct the rating configuration that caused the non-valid rating.
2. Configure the back-out pipeline. This pipeline uses the event extract output file as input. It backs out the existing balance impacts from discount accounts, aggregates the results, and outputs a back-out file.
3. Configure the rerating pipeline. This pipeline uses the back-out file as input. It rerates the events using the corrected rating configuration. The rerating process is the same as the rating process, however, the output file also includes the delta of the old and new balance impacts. Rated Event Loader loads the file into the BRM database.

Figure 60-2 shows the rerating steps described.

Figure 60-2 Pipeline Manager Rerating Process



About Loading Rerated Events into the BRM Database

You load rerated events into the BRM database by running the Rated Event Loader (RE Loader). RE Loader identifies events that have already been billed, ensuring that account balances are accurately updated. See "Adjusting Account Balances".

To load rerated events into the BRM database, you perform the following steps:

1. Set up the RE Loader processing directories.
2. Configure RE Loader to load the rerated event types by configuring the RELoader_home**/Infranet.properties** file.
3. Run RE Loader.

Adjusting Account Balances

RE Loader determines if each event has been billed. For events that *have* been billed, RE Loader creates an adjustment event (**/item/adjustment**) with the appropriate balance impact and sets the event type to **/event/billing/adjustment/event**. It then loads the events and updates the account balances. RE Loader updates the event as having been rerated and keeps a history of each update.

About Back-Out-Only Rerating

You use back-out-only rerating to reprocess events that were originally rated by Pipeline Manager. It enables you to back out only the event balance impacts without rerating the events.

For example, you might want to use back-out-only rerating when you find that call detail records (CDRs) rated and recorded in the BRM database contain incorrect values. In such cases, you can use back-out-only rerating to back out the event balance impacts without rerating the event.

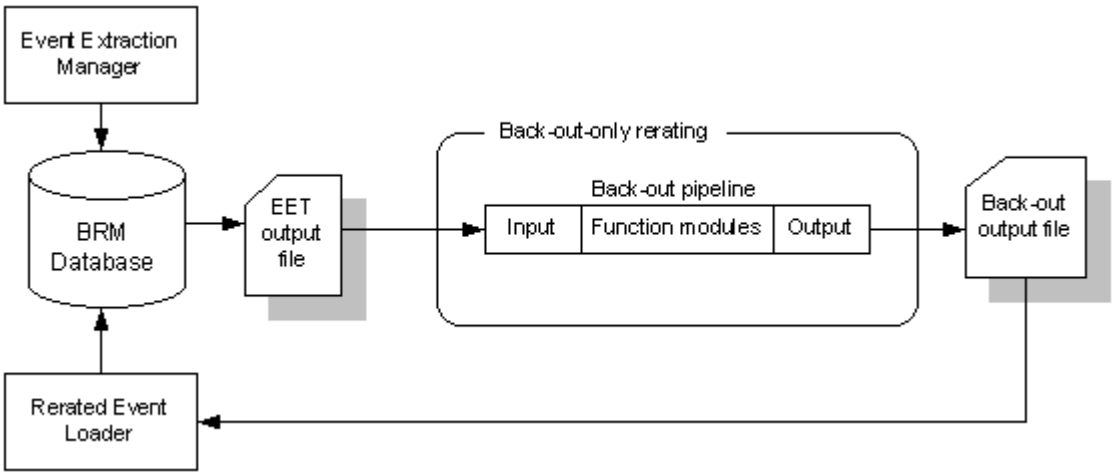
The output file created by backout-only rerating is loaded into the BRM database by the Rated Event (RE) Loader.

To process events for back-out-only rerating:

1. Use the Event Extraction Manager to extract the events from the BRM database. See "About Extracting Events for Back-Out-Only Rerating".
2. Use the backout pipeline to process the output file generated by the Event Extraction Manager. See "About Configuring the Backout Pipeline for Back-Out-Only Rerating".
3. Use the Rated Event Loader to load the output file generated by the backout pipeline into the BRM database. See "About Loading Rerated Events into the BRM Database".

Figure 60-3 shows the back-out-only rerating process:

Figure 60-3 Back-Out-Only Rerating Process



About Extracting Events for Back-Out-Only Rerating

Use the Event Extraction Manager to find and extract events from the BRM database for backout-only processing. The Event Extract Tool (**pin_event_extract**) generates an output file containing the events in EDR format.

To extract events for back-out-only rerating, run **pin_event_extract** with the **-e** parameter.

> **Note:**
>
> If you do not use the **-e** parameter, Pipeline Manager sends the EDRs in the output file to the rerating pipeline for rerating.

About Configuring the Backout Pipeline for Back-Out-Only Rerating

You use the backout pipeline to process the EDRs extracted by the Event Extraction Manager. The backout pipeline backs out the event balance impacts and writes the events to an output file.

The backout pipeline generates output files for both normal rerating and back-out-only rerating:

- In normal rerating, the output file is processed by the batch-rerating pipeline.
- In back-out-only rerating, the output file is processed by RE Loader.

Both types of rerating require different file formats and output grammar.

RE Loader loads events of different types from separate directories. For this reason, with back-out-only rerating, the backout pipeline sends the EDRs to different output streams based on the event type. For example, the backout pipeline writes all GSM events in one file, and all GPRS events in another file, and sends them to separate output directories.

To configure back-out-only rerating:

1. Configure the OUT_GenericStream module to generate the output file using the output grammar for back-out-only rerating and to send the output files to different directories based on the event type. See "Configuring the OUT_GenericStream Module".
2. Configure RE Loader to load the back-out output file into the BRM database. See "About Loading Rerated Events into the BRM Database".

Configuring the OUT_GenericStream Module

The OUT_GenericStream module formats the output for back-out-only rerating based on the grammar file specified in its registry. The EXT_OutFileManager sends the data to the output directories based on the event type specified in its registry. You configure the EXT_OutFileManager module inside the OUT_GenericStream module.

To configure the OUT_GenericStream and EXT_OutFileManager modules:

- Set the **Grammar** entry to

  `./formatDesc/Formats/Solution42/V670_EVENT_LOADER_OutGrammar.dsc.`

- Set the **OutputPath** entry to specify the output directory. For example, for GSM events, set **OutputPath** to

  `./data/backout/out/gsm/telephony.`

Sample OUT_GenericStream registry for GSM events:

```
BackOutOnlyTELOutput
{
     ModuleName = OUT_GenericStream
     ProcessType = ALL_BCKOUT
     EventType = /event/delayed/session/telco/gsm
     Module
     {
          Grammar = ./formatDesc/Formats/Solution42/V670_EVENT_LOADER_OutGrammar.dsc
          DeleteEmptyStream = True
          OutputStream
          {
               ModuleName = EXT_OutFileManager
               Module
               {
                    OutputPath = ./data/backout/out/gsm/telephony
                    OutputPrefix = test_TEL
                    OutputSuffix = .out
                    TempPrefix   = .
                    TempDataPath    = ./data/backout/out/gsm/telephony
                    TempDataPrefix  = tel.tmp.
                    TempDataSuffix  = .data
                    Replace = TRUE
               }
          }
     }
}
```

Sample OUT_GenericStream registry for GPRS event:

```
BackOutOnlyGPRSOutput
{
     ModuleName = OUT_GenericStream
     ProcessType = ALL_BCKOUT
     EventType = /event/delayed/session/gprs
     Module
     {
          Grammar = ./formatDesc/Formats/Solution42/V670_EVENT_LOADER_OutGrammar.dsc
          DeleteEmptyStream = True #defaults to true
          OutputStream
          {
               ModuleName = EXT_OutFileManager
               Module
               {
                    OutputPath = ./samples/wireless/data/backout/out/gprs
                    OutputPrefix = test_GPRS
                    OutputSuffix = .out
                    TempPrefix   = .
                    TempDataPath    = ./samples/wireless/data/backout/out/gprs
                    TempDataPrefix  = gprs.tmp.
                    TempDataSuffix  = .data
                    Replace = TRUE
               }
          }
     }
} # end of BackOutOnlyGPRSOutput
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

About Synchronizing Rating and Loading Applications

When you use Pipeline Manager to rerate wireless events and load those events using RE Loader, you must synchronize the following applications:

- Pipeline Manager
- Event Extraction
- Rated Event Loader

You must synchronize these applications for two reasons:

- RE Loader locks the BRM database tables that it loads into, making them inaccessible to other applications during the loading process. Event Extraction Manager returns an error if it cannot access the database tables.
- If any events have not been rated and loaded before Pipeline Manager starts rerating, the account balances can be incorrectly updated when those events are processed after rerating is complete.

You must synchronize these applications manually by stopping and restarting them in a specific order:

1. Ensure that all events have been rated by Pipeline Manager.
2. Ensure that all rated events have been loaded by RE Loader.

There should be no remaining unrated files in the pipeline output directories. If errors occurred during loading that require you to reload events, perform those tasks before continuing.

3. Stop the rerating pipeline.
4. Stop the Batch Controller, thereby disabling any scheduled RE Loader processes.
5. Run the Event Extraction utility.
6. Start the backout pipeline to remove all the erroneous impact balances from the discount accounts.
7. Stop the backout pipeline and restart the rerating pipeline to rerate the events. The rerating pipeline can be restarted using the Pipeline Manager EventHandler.
8. If you configured RE Loader to run *manually*, run the RE Loader **pin_rel** utility. If errors occur during loading, correct them and reload the events before continuing.
9. If you configured RE Loader to run *automatically*, enable the REL handler that runs **pin_rel** in the Batch Controller configuration file by doing the following:
   - Stop and restart the Batch Controller to run **pin_rel** and load the rerated events. If errors occur during loading, correct them and reload the events before continuing.
   - Disable the REL handler that runs **pin_rerel** in the Batch Controller configuration file.
10. Enable RE Loader in the Batch Controller configuration file.

61

Using Event Extraction Manager

This chapter describes how to use Oracle Communications Billing and Revenue Management (BRM) Event Extraction Manager to extract prerated events for rerating.

> **Note:**
>
> Event Extraction Manager is packaged with Rated Event (RE) Loader.

Before extracting events, you must know the following:

- Basic BRM concepts
- BRM system architecture
- How to create and edit BRM configuration files

About Event Extraction Manager

Event Extraction Manager is the first application used in the rerating process. You use it to find and extract events from the BRM database that must be rerated by Pipeline Manager and then reloaded into the BRM database by Rated Event (RE) Loader.

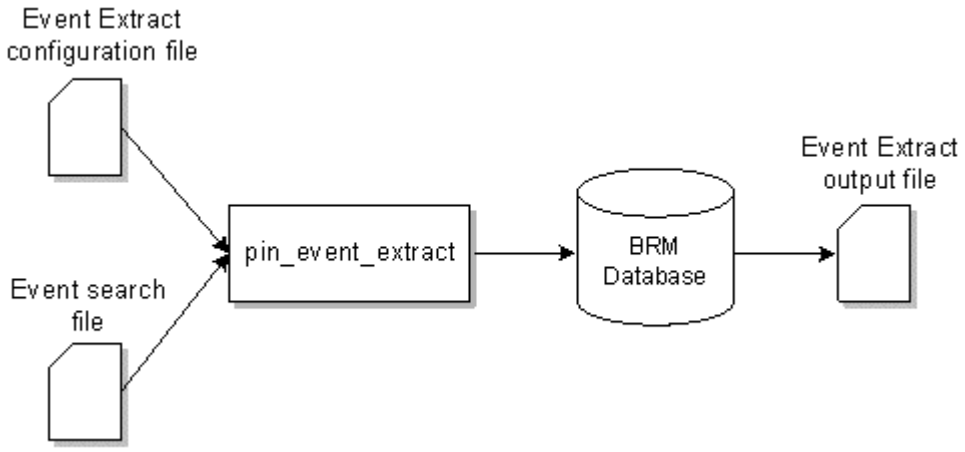
Event Extraction Manager does not modify or process events, nor does it lock accounts in your database. This enables BRM and other applications to safely access accounts in the database while you are extracting events.

Event Extraction Manager consists of the following components:

- The event extract configuration file (**pin.conf**), which specifies how the **pin_event_extract** utility connects to your BRM system and the size and name of the event extract output file
- The **pin_event_extract.cfg** file, which specifies the event search criteria
- The **pin_event_extract** utility, which searches the BRM database for events meeting the search criteria and writes the results to an output file
- The event extract output file, which contains the list of events to rerate

Figure 61-1 shows Event Extraction Manager:

Figure 61-1 Event Extraction Manager



About the Event Extract Configuration File

The event extract configuration file (**pin.conf**), which specifies how the **pin_event_extract** utility connects to your BRM system, and the size and name of your event extract output file.

If you are extracting events from a multischema system, you must modify this file for each database schema in your system. For information, see "Extracting Events from a Multischema System".

For information on how to create this file, see "Configuring Connection and Output File Parameters".

About the pin_event_extract.cfg File

The event search file (*BRM_home*/**apps/pin_event_extract/pin_event_extract.cfg**) specifies the criteria that the **pin_event_extract** utility uses to search for events in your BRM database. You can search for events based on a wide variety of criteria, including account number, charge offer name, and impact category.

For information about the event search criteria, see "Event Search Criteria".

For information on creating a **pin_event_extract.cfg** file, see "Specifying Which Events to Extract for Rerating".

About the pin_event_extract Utility

The **pin_event_extract** utility builds a search query with the criteria specified in the **pin_event_extract.cfg** file and then run it against the specified database schema. The search query selects only events that have a balance impact type of pipeline-prerated, indicating that Pipeline Manager has already rated the events.

About the Event Extract Output File

The **pin_event_extract** utility prints to a file the list of events meeting the search criteria. Results are written in tab-delimited columns conforming to the Oracle CDR format. You send this file directly to Pipeline Manager for rerating.

The default implementation defines the event-field-to-output file mapping for GSM and GPRS events. However, you can create additional categories by modifying the PCM_OP_BILL_POL_CONFIG_EET policy opcode.

When the number of events meeting the search criteria exceeds the specified maximum file size or number of EDRs, the utility generates multiple output files and appends a number, such as **_1**, **_2**, and so on, to each file name.

When you use the sequence number search criteria, the utility automatically appends a sequence number to the file name and ignores the specified maximum file size and number of EDRs.

Event Search Criteria

Event Extraction Manager searches for events by using the criteria in Table 61-1.

To specify which events to extract:

- To extract events generated for a specific brand, run the **pin_event_extract** utility with the **-b** *BrandName* option.
- To extract events with a specified starting timestamp, run the **pin_event_extract** utility with both the **-f** *ConfigFileName* and **-s** options.
- To extract events using all other criteria, edit the **pin_event_extract.cfg** file.

> **Note:**
>
> The event start time, event end time, and event type criteria are required.

Table 61-1 Event Search Criteria

Criteria	Entry Name	Description	Required
account number	`account`	Extracts events generated by the specified account. **Caution:** If you use this criteria, ensure the following: – All events for the specified accounts are available for rerating – All specified accounts reside in the same database schema – The event start time, event end time, and event type criteria are required.	no

Table 61-1 (Cont.) Event Search Criteria

Criteria	Entry Name	Description	Required
deal name	`deal`	Extracts events generated by all charge offers and discount offers associated with the specified bundle. **Caution:** Do not use this criteria with the charge offer or charge search criteria. Doing so prevents the utility from extracting all events associated with the specified bundle.	no
event occurrence start and end times	`event_start_time_stamp` `event_end_time_stamp`	Extracts events that occurred after the specified starting time stamp and before the specified ending time stamp. The start and end times are inclusive.	yes
event creation start and end times	`event_created_start_time_stamp` `event_created_end_time_stamp`	Extracts events loaded into the database after the specified creation start time stamp and before the specified creation end time stamp. The start and end times are inclusive.	no
event type	`event_category`	Extracts the specified event type. By default, this criteria is set to extract the **/event/delayed/session/telco/gsm** event type.	yes
G/L ID number	`gl_id`	Extracts events with the specified general ledger G/L ID.	no
impact category name	`impact_category`	Extracts events rated with the specified impact category.	no
item object number	`item_obj`	Extracts events generated for the specified item.	no
product name	`product`	Extracts events generated for the specified charge offers. **Caution:** Do not use this criteria with the bundle or charge search criteria.	no
charge name	`rate_plan`	Extracts events rated by the specified charge. **Caution:** Do not use this criteria with the bundle or charge offer search criteria.	no

Table 61-1 (Cont.) Event Search Criteria

Criteria	Entry Name	Description	Required
resource ID	`resouce_id`	Extracts events with the specified balance element ID. This criteria works best when the balance element ID represents a noncurrency asset such as Internet hours.	no
search event field	`search_event_type` `search_field` `search_parameter`	Extracts events of type *search_event_type* where *search_field* contains *search_parameter*. For example, you can use this search criteria to extract events based on custom fields. – *search_event_type* specifies the type of event to search. Only events of this type or a subclass of this event type are extracted **Important:** All events of type *search_event_type* must contain the *search_field* . You must ensure that the storable class specified by the event type contains *search_field*. – *search_field* specifies the field in the event used for event extraction. Only those events where *search_field* is equal to *search_parameter* are extracted. – *search_parameter* specifies the value of *search_field*. **Note:** *search_parameter* is used in addition to other search criteria specified in the **pin_event_extract.cfg** file.	no

Table 61-1 (Cont.) Event Search Criteria

Criteria	Entry Name	Description	Required
sequence number	`file_sequence_number`	Extracts events processed with the specified file sequence number. This number is assigned by Pipeline Manager and then added to the event object when RE Loader loads the event into the database. This enables you to extract and rerate events that were processed at a particular time. You can find an event's file sequence number in the **/batch/rel** object. When you use this criteria, the **pin_event_extract** utility automatically appends the file sequence number to the output file name. **Note:** You can specify any sequence number other than 0. RE Loader uses 0 to indicate that Pipeline Manager did not assign a sequence number to the event.	no
service name	`service`	Extracts events generated for the specified service. **Important:** The specified service must correspond to a unique service type.	no

Synchronizing Extraction and Rating Applications

It is important to rate all events before running Event Extraction Manager. If there are events not yet rated when you run Event Extraction Manager, your account balances will not be correct.

Event Extraction Manager and RE Loader use the same status table, REL_EVENT_EXTRACT_SYNC_T, to check if one of the other applications is running. If you attempt to start Event Extraction Manager while RE Loader is running, Event Extraction Manager fails to start and returns an error message.

However, if Event Extraction Manager or RE Loader terminate abnormally, the status table may be incorrectly left in a locked state. You must use the **pin_event_extract** utility's override option before you can extract events.

Extracting Events

Extracting events includes the following tasks:

Configuring Connection and Output File Parameters

To configure your connection and output file parameters:

1. Open the event extract configuration file (*BRM_home*/**apps/pin_event_extract/pin.conf**) in a text editor.
2. Edit the connection and log file entries.
3. Specify the EDR output file name:

   - pin_event_extract filename SOL42_*SenderReceiver*

   where *Sender* is the code for the sender of the file, and *Receiver* is the code for the recipient of the file. For example, if the sender's code is D00D1 and the receiver's code is SOL42, replace *SenderReceiver* with **D00D1SOL42**.

   > **Note:**
   >
   > - When you use the sequence number search criteria, the utility automatically appends the file sequence number to the file name.
   > - When the output exceeds the specified file maximums, the utility automatically generates multiple output files and appends a number to each file name.

4. Specify the UTC Time Offset: where UTC Time Offset is the difference between the local time and UTC time:

   - pin_event_extract UTCoffset *Value*

5. Specify the specification version number:

   - pin_event_extract specvernum *VersionNumber*

6. Specify the specification release version number:

   - pin_event_extract specrelver *ReleaseNumber*

7. Specify the maximum number of EDR creation threads:

   - pin_event_extract num_threads *MaximumNumber*

8. Specify the number of EDR records Event Extraction Manager retrieves during each step search:

   - pin_event_extract step_size *SearchNumber*

9. Specify the maximum size, in bytes, of the event extract output file:

   - pin_event_extract maxfilesize *FileSize*

When the file size exceeds the specified maximum, the **pin_event_extract** utility generates multiple output files and appends a number, such as **_1**, **_2**, and so on, to each file name.

10. Specify the maximum number of EDRs allowed in each output file.

```
- pin_event_extract    maxEDRs    MaximumEDRs
```

When the number of EDRs exceeds the specified maximum, the **pin_event_extract** utility generates multiple output files and appends a number, such as **_1**, **_2**, and so on, to each file name.

11. Save and close the file.

Specifying Which Events to Extract for Rerating

The **pin_event_extract** utility finds the events for rerating by creating a search query. You specify the parameters for the search in the *BRM_home***/apps/pin_event_extract/pin_event_extract.cfg** file.

To extract events:

1. Open the *BRM_home***/apps/pin_event_extract/pin_event_extract.cfg** file in a text editor.
2. Set the **event_start_time_stamp** and **event_end_time_stamp** entries.

```
event_start_time_stamp MM/DD/YYYY [hh:mm:ss]
event_end_time_stamp MM/DD/YYYY [hh:mm:ss]
```

where *hh:mm:ss* is the time in 24-hour mode. For example, enter **16:00:00** to specify 4:00 p.m. If you do not specify *hh*:*mm*:*ss*, the time defaults to midnight (00:00:00).

The **pin_event_extract** utility extracts events that occurred between the specified start date and end date (inclusive).

3. Set the **event_category** entry:

```
event_category      category
category            EventTypeName
```

where:

- *category* specifies the type of event to extract. The default implementation includes the **gsm** and **gprs** event categories only, but you can create additional categories by modifying the PCM_OP_SUBSCRIPTION_POL_CONFIG_EET policy opcode.
- *EventTypeName* specifies the name of the event you want extracted. For example, **/event/delayed/activity/gprs**.

> **Note:**
>
> You can list multiple event types on one line by using colons (**:**) or semicolons (**;**) to delimit events.

4. Set your search criteria. The **pin_event_extract.cfg** file includes examples and instructions.

For an overview, see "Event Search Criteria".

5. Save and close the file.

Extracting Events from Specific Partitions

Events are loaded into database tables that are partitioned by a period such as days, weeks, or months. You can improve event extraction performance by limiting the number of partitions from which events are extracted. To do this, specify the start and end dates of when the events were loaded into the database.

For example, if your database tables are partitioned by months, you can extract events from a single partition by specifying the start and end day of a single month.

To extract events from specific partitions:

1. Open the *BRM_home*/**apps/pin_event_extract/pin_event_extract.cfg** file in a text editor.
2. Set the **event_created_start_time_stamp** and **event_created_end_time_stamp** entries. The **pin_event_extract** utility extracts events that were loaded into the database on or after the start time and on or before the end time.

   ```
   event_created_start_time_stamp MM/DD/YYYY hh:mm:ss
   event_created_end_time_stamp MM/DD/YYYY hh:mm:ss
   ```

   where:

   - *MM/DD/YYYY* is the month, day, and year.
   - *hh:mm:ss* is the time in 24-hour mode. For example, enter *16:00:00* to specify 4:00 p.m.

   > **Note:**
   >
   > Entering only a day returns an error.

Sample pin_event_extract.cfg File for a Specific Charge

The following sample **pin_event_extract.cfg** file extracts events that meet the following criteria:

- Created between 6:00 p.m. on June 1, 2002, and 6:00 p.m. on August 30, 2002.
- Is the following prerated GSM event: **/event/delayed/session/telco/gsm**.
- Rated with the Free Saturdays charge.

```
event_start_time_stamp 06/01/2002 18:00:00
event_end_time_stamp 08/30/2002 18:00:00

event_category   gsm
gsm              /event/delayed/session/telco/gsm

rate_plan Free Saturdays
```

Sample pin_event_extract.cfg File for a Specific Sequence Number

The following sample **pin_event_extract.cfg** file extracts events that meet the following criteria:

- Created between 10:00 a.m. on January 1, 2002 and 9:59:59 a.m. on January 31, 2002.

- Represents a prerated GPRS event
- Processed with the 777888999 sequence number

```
event_start_time_stamp 01/01/2002 10:00:00
event_end_time_stamp 01/31/2002 09:59:59
event_type /event/delayed/session/gprs
file_sequence_number 777888999
```

Running the pin_event_extract Utility

The **pin_event_extract** utility generates a search query with your specified search criteria. See "pin_event_extract" for more information.

To extract events:

1. Ensure that all events that you want to extract have been processed.
2. Ensure that RE Loader is not running.

   For information, see "Synchronizing Extraction and Rating Applications".

   > **Note:**
   >
   > If RE Loader and **pin_event_extract** are run simultaneously, **pin_event_extract** extracts only those records that are completely processed by RE Loader.

3. Run the **pin_event_extract** utility:

   ```
   pin_event_extract -f ConfigFileName
   ```

   where *ConfigFileName* is the name and location of the **pin_event_extract.cfg** file.

Troubleshooting Event Extraction Errors

Event extraction may fail for several reasons:

- RE Loader is currently running, preventing Event Extraction Manager from starting. Wait for RE Loader to finish loading events before restarting Event Extraction Manager.
- The status table was incorrectly locked, preventing Event Extraction Manager from starting. This can occur when one of the applications terminates abnormally and cannot reset the status table before exiting. To reset the status table:

  > **Note:**
  >
  > Do not use this option unless you are certain that RE Loader is stopped.

  ```
  pin_event_extract -o TRUE
  ```

You can check the event extract log file, created by default in the *BRM_home***/var/pin_event_extract** directory, for information on why extraction failed.

After you fix the problem, you can safely rerun the **pin_event_extract** utility with the same **pin_event_extract.cfg** file.

Sending the Output File to Pipeline Manager

To begin rerating events, send the event extract output file to Pipeline Manager for processing.

Extracting Events from a Multischema System

To extract events from a multischema system:

1. On the system on which Event Extraction Manager is installed, open the *BRM_home*/**apps/pin_event_extract/pin.conf**in a text editor.
2. Specify which events to extract by using the **pin_event_extract.cfg** file.
3. Run the **pin_event_extract** utility and fix any errors.
4. Change the connection and output file parameters for the next schema.
5. Run the **pin_event_extract** utility and fix any errors.
6. Repeat steps 4 and 5 until you have extracted events from all schemas.

Customizing How to Extract Events for Rerating

You can use the PCM_OP_SUBSCRIPTION_POL_CONFIG_EET policy opcode to customize which event fields are extracted for rerating; for example, you can modify which GPRS fields are extracted, or you can define the mapping for a custom event category.

The PCM_OP_SUBSCRIPTION_POL_CONFIG_EET policy opcode defines, for each event category, the event fields that Event Extraction Manager passes to the event extract output file. The default implementation defines the event-field-to-output-file mapping for GSM and GPRS event categories only.

This policy opcode is called by Event Extraction Manager worker threads when the tool returns a list of POIDs that meet the search criteria.

By default, the PCM_OP_SUBSCRIPTION_POL_CONFIG_EET policy opcode takes an event's POID and category name as input and performs the following functions:

- Reads the entire event from the BRM database
- Maps predefined fields for GSM and GPRS events to a record buffer in the order received

> **Note:**
>
> The **pin_event_extract** utility writes the contents of the record buffer to the event extract output file.

You can customize the PCM_OP_SUBSCRIPTION_POL_CONFIG_EET policy opcode to pass extended fields to the event extract output file. For example, you can modify which GPRS fields are passed to the output file, or you can define the mapping for a custom event category.

You can do the following:

- Read any additional event data based on the given POID
- Handle any new input formats: the policy opcode is written for the Oracle CDR forma
- Define the event-field-to-output-file mapping for any new categories

- Write data to the correct position in the record buffer

62
Configuring Rerating in Pipeline Manager

This chapter describes how to rerate events by using the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

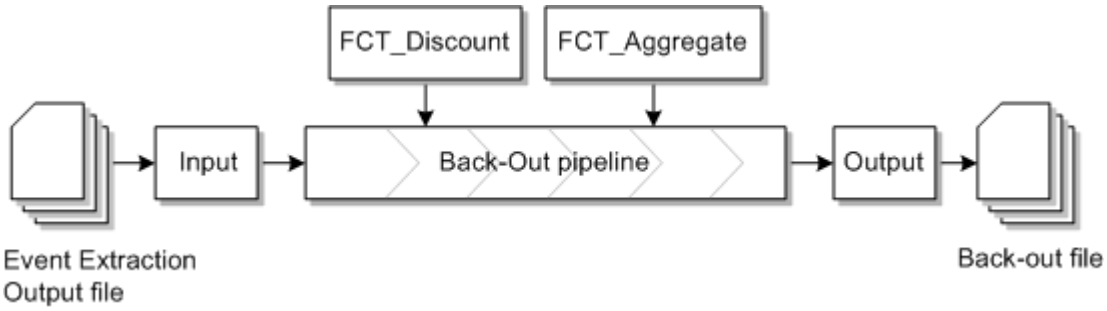
About the Back-Out Pipeline

The back-out pipeline is a separate pipeline and contains the following modules:

- The FCT_Discount module removes the EDR discount balance impacts from the discount accounts.
- The FCT_AggreGate module calculates the aggregation results and writes them to the back-out result file. It writes one result file for each transaction.

Figure 62-1 shows the back-out pipeline:

Figure 62-1 Back-Out Pipeline



Configuring the Back-Out Pipeline

When you configure the back-out pipeline, you configure the following parameters in the pipeline registry:

- The ISC_AddCBD iScript. You configure this by configuring an instance of the FCT_IScript module.
- The FCT_Discount module. You set the **BackOut** registry entry to TRUE.
- The FCT_AggreGate module. You set the **BackOut** registry entry to TRUE.

Pipeline Manager creates one control file and one result file for each transaction and each aggregation scenario.

About Starting the Back-Out Pipeline

Before you start the back-out pipeline stop all rating pipelines and ensure that there are no outstanding events being loaded.

You start the back-out pipeline by sending a semaphore:

```
ifw.Pipelines.ALL_BCKOUT.Active = TRUE
```

About Stopping the Back-Out Pipeline

You stop the back-out pipeline by sending a semaphore:

```
ifw.Pipelines.ALL_BCKOUT.Active = FALSE
```

About the Rerating Pipeline

The rerating pipeline contains the same function modules as the rating pipeline, except that it does not contain the FCT_CallAssembling and FCT_DuplicateCheck modules. The rerating pipeline locks the appropriate customer accounts until rerating is completed.

> **Note:**
>
> Before you start the rerating pipeline, ensure that you corrected all configuration data in the Pipeline Manager database, such as the charge that caused the incorrect calculation.

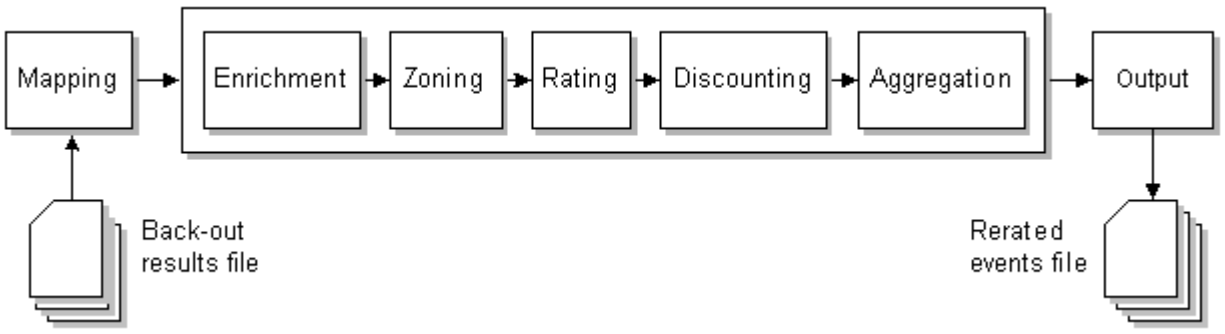
After the EDRs are re-rated with the updated configuration data, the FCT_BillingRecord module writes to the BRM billing record the difference between the old and re-rated balance impacts for each EDR.

The rerating output file contains all events for which the price has changed as a result of rerating and the change in the balance impact.

The output module sends the output file with the updated impact balances to the Rerated Event Loader (RREL) which loads them into the BRM database.

Figure 62-2 shows the rerating pipeline:

Figure 62-2 Rerating Pipeline



About Configuring the Rerating Pipeline

When you configure the rerating pipeline, you configure the same modules in the registry as for the normal rating pipeline. The only difference is that you do not configure the FCT_DuplicateCheck and the FCT_CallAssembling modules.

For the rerating pipeline, you use special backout input and output grammar files, and you create separate input and output directories.

For a sample registry for the rerating pipeline, see the **rerating.reg** file.

This sample shows the input configuration for a rerating pipeline. It uses a special backout grammar file, and a separate input directory.

```
Input
  {
    UnitsPerTransaction = 1
      InputModule
      {
        ModuleName  = INP_GenericStream
        Module
        {
          DefaultOutput = RerateOutput
          Grammar = ./formatDesc/Formats/Solution42/SOL42_V630_REL_InGrammar_BACKOUT.dsc
          InputStream
          {
            ModuleName = EXT_InFileManager
            Module
            {
              InputDirEmptyTimeout = 10
              InputPath   = ./samples/wireless/data/backout/in
              InputPrefix = test.
              InputSuffix = .edr
              DonePath    = ./samples/wireless/data/backout/done
              DonePrefix  = test.
              DoneSuffix  = .done
              ErrorPath   = ./samples/wireless/data/backout/err
              ErrorPrefix = test.
              ErrorSuffix = .err
              TempPrefix  = tmp.
              Replace     = TRUE
            }
          } # end of InputStream
        }
      } # end of InputModule
    } # end of Input
```

This sample shows the output for telephony EDRs in a rerating pipeline. It uses a special backout output grammar file and a separate output directory.

```
TELOutput
   {
     ModuleName = OUT_GenericStream
     Module
     {
       Grammar = ./formatDesc/Formats/Solution42/SOL42_V630_REL_OutGrammar.dsc
       DeleteEmptyStream = True
      OutputStream
       {
         ModuleName = EXT_OutFileManager
         Module
         {
           OutputPath = ./samples/wireless/data/rerate/telout
           OutputPrefix = test
           OutputSuffix = .out
           TempPrefix   = .
          TempDataPath    = ./samples/wireless/data/rerate/telout
           TempDataPrefix  = tel.tmp.
           TempDataSuffix  = .data
```

```
          Replace = TRUE
        }
      }
    }
  } # end of TELOutput
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

About Starting the Rerating Pipeline

Before you start the rerating pipeline, ensure that the back-out pipeline has processed all EDRs that were extracted by the Event Extraction Tool. Also ensure that all other rating pipelines are stopped.

To start the rerating pipeline, use the event handler to start the rerating pipeline when the input directory is empty. To do so, configure the corresponding section in the registry. The EVT_INPUT_DIR_EMPTY event is triggered by the EXT_InFileManager when the input directory is emptied. In this sample, the event starts the rerating pipeline.

```
EventHandler
{
  ModuleName = EVT
  Module
  {
    Events
    ifw.Pipelines.ALL_BCKOUT.Input.InputModule.Module.InputStream.Module
    {
      EVT_INPUT_DIR_EMPTY = ./start_rerate.sh
    }
  }

  Buffer
  {
    Size = 100
  }
 }
}
```

To define when to send EVT_INPUT_DIR_EMPTY event, use the EXT_InFileManager **InputDirEmptyTimeout** registry entry.

About Stopping the Rerating Pipeline

You stop the rerating pipeline manually by sending a semaphore:

```
ifw.Pipelines.RERATE.Active = FALSE
```

63

About Comprehensive Rerating Using pin_rerate

This chapter provides an overview of rerating using the Oracle Communications Billing and Revenue Management (BRM) **pin_rerate** utility to rerate real-time-rated events and pipeline-batch-rated events.

> **Note:**
>
> This documentation does not apply to rerating only pipeline-batch-rated events by using a batch rerating pipeline.

About Comprehensive Rerating

The **pin_rerate** utility provides a comprehensive rerating solution: You can rerate both real-time-rated and pipeline-batch-rated events concurrently or, if you do not use Pipeline Manager for batch rating, you can rerate only real-time-rated events. You can select accounts and events for rerating based on any event criteria. And you can configure BRM to automatically rerate certain accounts and events when you run **pin_rerate**.

The **pin_rerate** utility controls the rerating workflow. It follows a different rerating process depending on whether batch pipeline rating is enabled. See:

- About Rerating Real-Time-Rated and Pipeline-Rated Events Concurrently
- About Rerating Events When You Do Not Use Pipeline Batch Rating

About Rerate Jobs

A rerate job is a set of objects in the BRM database used to track the accounts selected for rerating and the status of the rerating process. Rerate jobs are created for all accounts to be rerated by **pin_rerate**. (For more information about rerate job objects, see "How BRM Creates Rerate Jobs".)

Rerate jobs are typically associated with a batch of accounts, but can also be associated with a single account.

Rerate jobs are created manually when you use **pin_rerate** to select accounts for rerating. Rerate jobs are created automatically when certain events occur, such as when backdated events or rate changes trigger automatic rerating.

When you use **pin_rerate** to select accounts for rerating, the utility assigns accounts to each rerate job and creates 2 rerate jobs per transaction. This enables **pin_rerate** to more easily roll back a transaction if an error occurs.

You can configure the number of accounts assigned to each rerate job and the number of jobs to create per transaction.

About Rerating Real-Time-Rated and Pipeline-Rated Events Concurrently

When you use **pin_rerate** to rerate both real-time-rated and pipeline-batch-rated events concurrently, the following tasks are performed:

1. The utility retrieves the accounts for rerating from the BRM database based on search criteria that you specify.
2. BRM extracts the events associated with those accounts and then backs out the original event balance impacts.
3. BRM determines whether the original event was rated in real time or by the batch rating pipeline:
   a. It sends events previously rated in real time to the real-time rating opcodes for rating and discounting.
   b. It sends events previously rated by a batch pipeline to the real-time rerating pipeline for rating and discounting. See "About the Real-Time Rerating Pipeline".
4. BRM records the rerated events in the BRM database.

About the Real-Time Rerating Pipeline

The **pin_rerate** utility sends events previously rated by the batch rating pipeline to a real-time rerating pipeline to be rerated.

The **pin_rerate** utility routes the pipeline-rated events to the Connection Manager (CM). The CM sends the pipeline-rated events in the form of an flist to the NET_EM pipeline module. The NET_EM module transfers the event to the real-time rerating pipeline for rerating.

The real-time rating opcodes add to the flist all the enrichment data needed by the real-time rerating pipeline to rate the event. For example, if the real-time rating opcodes determine that a discount should be applied, it adds the discount information to the flist.

> **Note:**
>
> Real-time rerating process does not check credit limits when applying discounts. To configure rerating to check credit limits, create a custom iScript that sets the DETAIL.CREDIT_LIMIT_CHECK EDR field to **1**.

Similar to a batch rerating pipeline, a real-time rerating pipeline performs both rating and discounting functions.

To rerate events, the real-time rerating pipeline gets the new pricing information from the Pipeline Manager database. Certain configuration data, such as currency and noncurrency balance element information, are obtained from the BRM database.

The real-time rerating pipeline performs the following tasks:

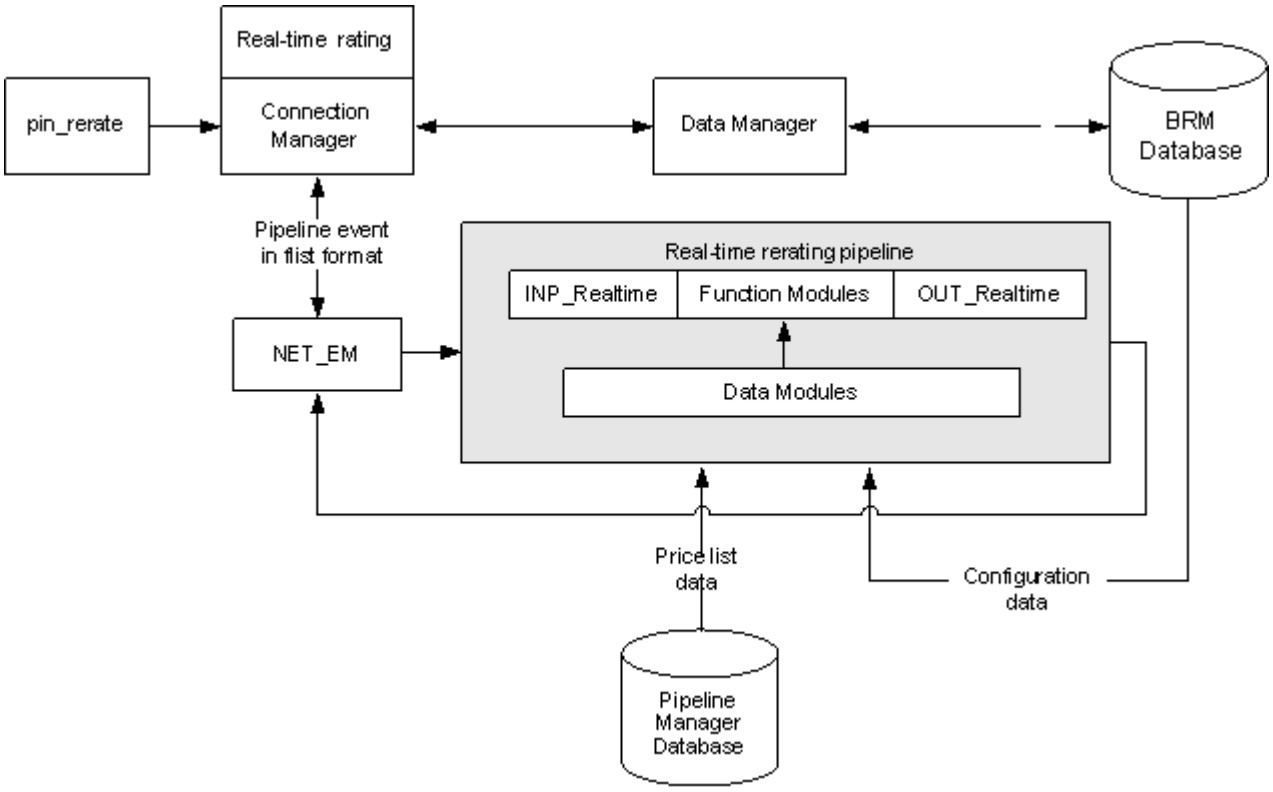
1. The INP_Realtime input module converts the flist received from the NET_EM module to event data record (EDR) format and creates an EDR container.

> **Note:**
>
> You can also create and run an iScript to manipulate data in the EDR container before it is sent to the pipeline modules.

2. The pipeline function modules calculate the new balance impacts by using the new pricing and discounting information and then adds the balance impacts to the EDR container.
3. The OUT_Realtime module converts the EDR back into flist format and sends the flist to NET_EM.
4. NET_EM sends the flist with the new balance impacts back to the CM.
5. BRM updates the account's balances and records the event in the BRM database.

Figure 63-1 shows the data flow for rerating events by using the real-time rerating pipeline:

Figure 63-1 Real-Time Rerating Pipeline Data Flow



About Transaction Management for the Real-Time Rerating Pipeline

The real-time rerating pipeline does not use the Transaction Manager (TAM) module. Instead, transaction handling is provided by the CM. When a rerating transaction fails, the CM rolls back the transaction and restores all the account balances in the BRM database. For this reason, function modules and iScripts used by the real-time rerating pipeline are stateless.

> **Note:**
>
> If you use iScripts for custom processing, ensure that the iScripts are stateless.

How pin_rerate and the Batch-Rating Pipeline Synchronize Processes

While pipeline-rated events are being rerated, the batch pipeline temporarily suspends new EDRs for the accounts that are currently being rerated. When rerating is complete, account balance data in Pipeline Manager is synchronized with the account balance data in the BRM database, and the batch pipeline resumes processing the suspended EDRs using the updated data.

The batch pipeline and **pin_rerate** use the Account Synchronization DM and BRM standard recycling to synchronize processes.

When you run **pin_rerate** for concurrent rerating, the following actions are taken to synchronize **pin_rerate** and batch pipeline processes:

1. The **pin_rerate** utility retrieves the accounts to be rerated and creates the rerate jobs. The rerate jobs are queued until they are processed.
2. **pin_rerate** sends a business event through the Account Synchronization DM to notify the batch pipeline that rerating for the selected accounts is about to begin. When it receives the notification, the batch pipeline suspends rating incoming CDRs for the accounts and sends **pin_rerate** a corresponding acknowledgment event.
3. **pin_rerate** rerates the accounts in the rerate job and BRM records the new balance impacts of the rerated events in the BRM database.
4. **pin_rerate** sends a business event through the Account Synchronization DM to notify the batch pipeline that rerating is complete for the selected accounts. The batch pipeline synchronizes the account data in pipeline memory with the account data in the BRM database and resumes processing EDRs. The batch pipeline sends **pin_rerate** a corresponding acknowledgment event.
5. **pin_rerate** recycles the EDRs suspended during the rerating process by calling the standard recycling opcode.

After successful completion of each step, **pin_rerate** updates the rerate job status so that the next step in the rerate workflow can be run.

Rerate jobs that are processed during concurrent rerating can have the statuses shown in Table 63-1:

Table 63-1 Status for Rerate Jobs

Status	Description
NEW	The initial status of a rerate job.
WAITING_ACCOUNT_LOCKED	The status after **pin_rerate** has notified the batch pipeline to suspend EDRs for accounts in the rerate job.
ACCOUNT_LOCKED	The status after **pin_rerate** has received acknowledgment from the batch pipeline.
RERATED	The status after the events associated with the accounts in the rerate jobs have been rerated.

Table 63-1 (Cont.) Status for Rerate Jobs

Status	Description
READY_FOR_RECYCLE	The status after account data has been synchronized between the BRM database and Pipeline Manager and when the batch pipeline has resumed processing EDRs for the accounts in the rerate job.
COMPLETE	The status when rerating is completed for *all* the accounts in the rerate job.

About Suspending and Recycling EDRs during the Rerating Process

To suspend EDRs for Accounts that are being rerated, the pipeline DAT_AccountBatch module sets a rerate error code and a recycle key value in the EDR. The error code causes the EDR to be suspended by BRM standard recycling. The recycle key is used to select the EDRs to be recycled.

You must load suspended EDRs into the BRM database by running Suspended Event (SE) Loader. You typically schedule SE Loader to run automatically when you set up standard recycling.

Suspended EDRs are stored in the database until they are recycled. To recycle the EDRs that were suspended during the rerating process, you run **pin_rerate** with the **-process recycle** parameter after rerating is completed.

Procedure for Concurrent Rerating of Real-Time-Rated and Pipeline-Rated Events

To rerate both real-time-rated events and pipeline-batch-rated events concurrently, you perform the following steps:

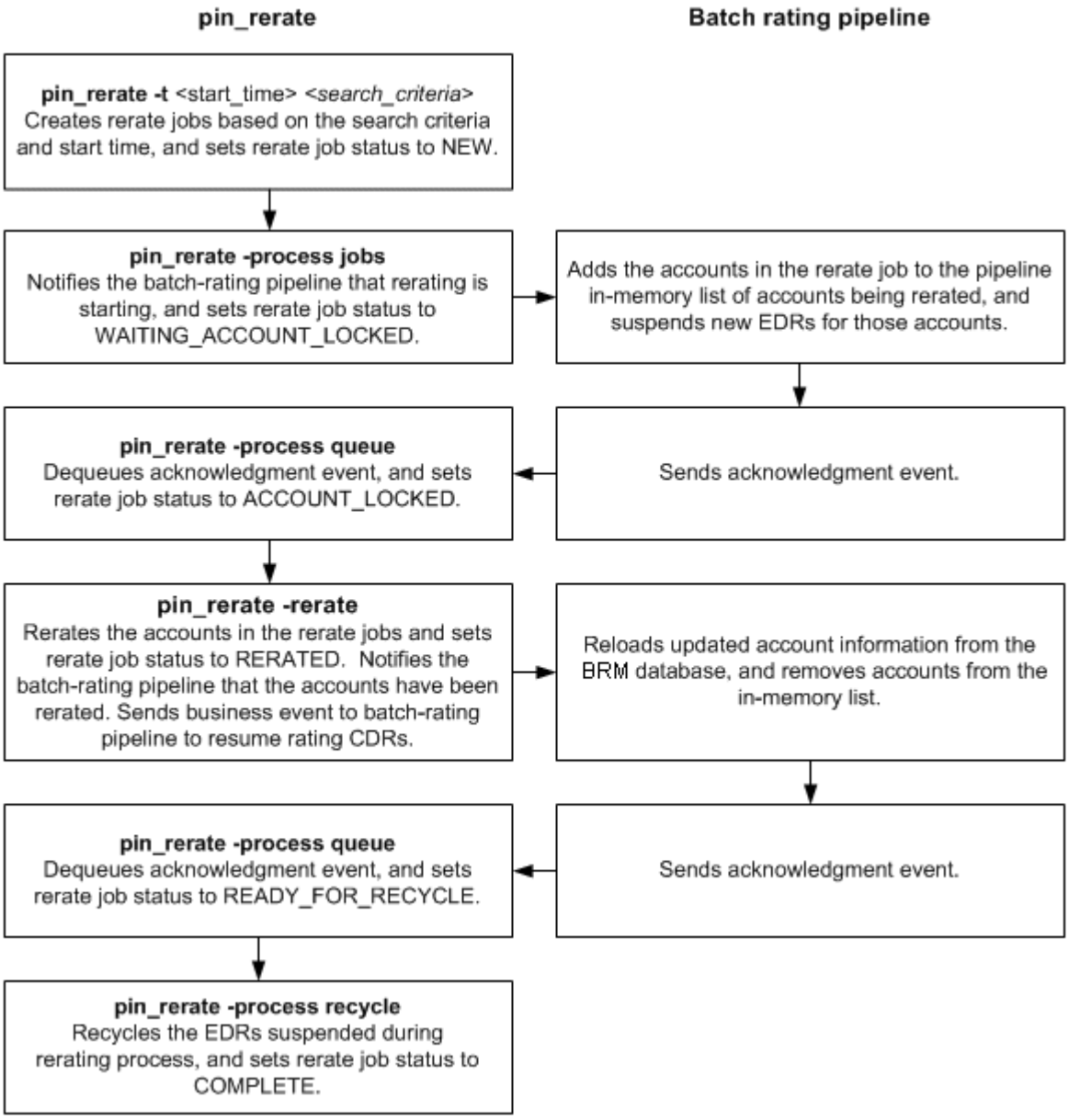
1. Run **pin_rerate** to select the accounts for rerating and create rerate jobs.

   > **Note:**
   >
   > Rerate jobs can also be created automatically through BRM automatic rerating or trigger-dependent rerating.

2. Run a series of **pin_rerate** commands to rerate the events associated with the accounts in the rerate job.
3. Run **pin_rerate** to recycle the EDRs associated with the accounts in the rerate job that were suspended during rerating.

Figure 63-2 shows the rerating commands, process flow, and job status transitions when you perform concurrent rerating:

Figure 63-2 Concurrent Rerating Process Flow



About Rerating Events When You Do Not Use Pipeline Batch Rating

If you do not use Pipeline Manager for batch rating, you can use **pin_rerate** to rerate only real-time-rated events.

> **Note:**
>
> Real-time-rated events are events that are rated by the BRM rating opcodes. Real-time-rated events include events that are loaded in batches by Universal Event (UE) Loader.

To rerate only real-time-rated events, you must configure the **batch_rating_pipeline** business parameter so that **pin_rerate** does not attempt to communicate with a batch pipeline, which will cause rerating to fail.

When rerating only real-time-rated events, the following actions are taken when you run **pin_rerate**:

1. The **pin_rerate** utility selects the accounts to be rerated from the BRM database based on the specified search criteria and creates rerate jobs.

   > **Note:**
   >
   > Rerate jobs can also be created automatically through BRM automatic rerating or trigger-dependent rerating.

2. **pin_rerate** invokes rerating of the selected accounts in the rerate jobs.
3. BRM extracts the events associated with the selected accounts from the BRM database and does the following:
   a. Rerates the real-time-rated events.
   b. Records the new balance impacts of the rerated events in the BRM database.

After the successful completion of each step, **pin_rerate** updates the rerate job status so that the next step in the rerate workflow can be run.

Rerate jobs that are processed when rerating only real-time-rated events can have the statuses shown in Table 63-2:

Table 63-2 Status for Real-Time Rerate Events

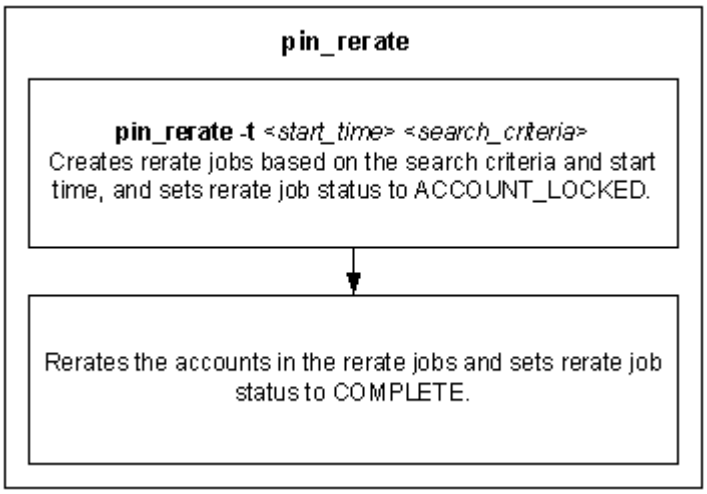
Status	Description
ACCOUNT_LOCKED	The initial status of rerate jobs.
COMPLETE	The status when rerating has completed for *all* the accounts in the rerate job.

Procedure for Rerating Only Real-Time-Rated Events

To rerate only real-time-rated events when you do not use Pipeline Manager for batch rating, you select the accounts to rerate and rerate the selected accounts using a single **pin_rerate** command.

Figure 63-3 shows the rerating command, process flow, and job status transitions when rerating only real-time-rated events:

Figure 63-3 Rerating Only Real-Time Events



How Failed Rerate Jobs Are Processed

The accounts in a rerate job are typically processed individually in separate rerate operations. When rerating fails for one or more accounts in a rerate job, **pin_rerate** sets the status of the accounts in the rerate job batch to FAILED. When rerating process is complete for the rerate job, **pin_rerate** creates a new rerate job consisting of only the accounts that failed. The new rerate job is processed the next time **pin_rerate** is run.

If an account selected for rerating is associated with a subscription service that was transferred during the period for which rerating is specified, then the account to which the service was transferred is included in the rerate job and all those accounts are rerated concurrently in a single rerate operation. When rerating fails for one of these accounts, then rerating fails for all accounts in the rerate request. In this case, **pin_rerate** creates a new rerate job containing all the accounts in the rerate request.

For example, subscription service X is originally owned by Account A and transferred to Account B on June 15. Later in the month, it is transferred from Account B to Account C. Rerating of Account A from June 1 also results in rerating of accounts B and C. Accounts A, B, and C are grouped together in a single rerate request. If rerating fails for any of these accounts, **pin_rerate** creates a new rerate job consisting of all three accounts in a single rerate request. The accounts are rerated again the next time **pin_rerate** is run.

About Automatic Rerating

When certain events occur in the BRM system (such as backdated purchase events), customer accounts require rerating so they are billed accurately. In automatic rerating, BRM automatically creates rerate jobs for these events.

> **Note:**
>
> Automatic rerating *does not* immediately rerate accounts when the rerate job is created. You must still run the **pin_rerate** utility.

Accounts and events associated with rerate jobs that are created automatically are rerated the next time you run **pin_rerate** to process rerate jobs: you do not need to first select the accounts and events by specifying **pin_rerate** selection criteria.

You enable automatic rerating by configuring event notification to call the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST policy opcode when events occur that require rerating.

Automatic rerating uses rerate reason codes. A rerate reason code is passed in to PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST, which uses the code to help determine whether rerating is required for an event.

BRM creates rerate jobs for automatic rerating as follows:

1. An event occurs that requires accounts to be rerated.
2. The opcode that triggers rerating when that particular event occurs creates a notification event.

   Table 63-3 summarizes the types of notification events used to trigger automatic rerating and the rerating scenario to which each event applies:

Table 63-3 Types of Notifications for Automatic Rerating

Notification Events for Automatic Rerating	Rerating Scenario
/event/notification/auto_rerate	Automatic rerating of backdated events.
/event/notification/rate_change	Automatic rerating for rate changes.
/event/notification/rollover_correction/rerate	Automatic rerating of rollover corrections due to delayed events.
An event by which you want rerating to be triggered	Automatic rerating for your custom rerating requirements.

3. The event notification mechanism calls PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST.
4. PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST takes as input the event type and the rerate reason code associated with the event and analyzes the event to determine if rerating is required.

> **Note:**
>
> BRM reserves the range of 100 to 120 for automatic rerating reason codes.

5. If rerating is required, PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST calls the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode along with the appropriate rerating criteria to create a rerate job.

PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST calls PCM_OP_SUBSCRIPTION_INSERT_RERATE_REQUEST only when the reason code passed in is one of those reserved for automatic rerating (reason codes between 100 and 120). By default, if a different reason code is passed in, the opcode does nothing. To create rerate jobs for reason codes other than those reserved for automatic rerating, you must customize PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST.

By default, PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST passes a reason code of **0** (no rerate reason) to PCM_OP_SUBSCRIPTION_INSERT_RERATE_REQUEST. To rerate accounts separately based on a rerate reason code, customize PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST to pass the rerate reason code it receives (or a different rerate reason code). The rerate reason code is stored in the rerate job object, and you can select rerate jobs based on the specific rerate reason code when you run **pin_rerate** to process rerate jobs.

> **Note:**
>
> You can also assign reason codes when you manually create rerate jobs by using **pin_rerate**.

You can also customize PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST in other ways. For example, you can change whether rerate jobs are created for specific events: such as when you want to rerate events for only a few (rather than all) automatic rerating scenarios.

How Rerating Affects Account Migration

When an account is selected for rerating, the account cannot be migrated to another database until rerating is complete. Otherwise, the account is not rerated. For this reason, the Account Migration Manager (AMM) does not allow migration of an account to another database if the account is being rerated by **pin_rerate**.

The AMM does not migrate an account if the account is in a rerate job and the rerate job status is one of the following:

- WAITING_ACCOUNT_LOCKED
- ACCOUNT_LOCKED
- RERATED
- READY_FOR_RECYCLE

However, the account is migrated if the rerate job status is NEW. In this case, the AMM deletes the account from the rerate job in the source database and creates a new rerate job with the account in the destination database.

> **Note:**
>
> After the account migration is complete, you must run **pin_rerate** in the destination database to rerate the account.

Managing Comprehensive Rerating with Custom Applications

BRM uses the following opcodes for comprehensive rerating with **pin_rerate**:

- PCM_OP_SUBSCRIPTION_RERATE_REBILL. See "How Comprehensive Rerating Works".

- PCM_OP_RERATE_INSERT_RERATE_REQUEST. See "How BRM Creates Rerate Jobs".
- PCM_OP_SUBSCRIPTION_PREP_RATE_CHANGE.
- PCM_OP_SUBSCRIPTION_RATE_CHANGE.
- PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE.

How Comprehensive Rerating Works

To rerate events, use PCM_OP_SUBSCRIPTION_RERATE_REBILL.

This opcode rerates the events for accounts identified by the **pin_rerate** utility, rerating one account at a time. The rerating start time is specified in the input flist. This opcode calls other opcodes to perform rerating functions.

PCM_OP_SUBSCRIPTION_RERATE_REBILL generates either a shadow event or an adjustment event to apply the results of rerating to the account balance in the current billing cycle:

- If the event is unbilled, a shadow event is created, which is an adjustment event that is applied to the bill item instead of the adjustment item.
- If the event is billed, or if the event is unbilled but already posted to the G/L, an adjustment event is created and applied to the adjustment item.

PCM_OP_SUBSCRIPTION_RERATE_REBILL does not rerate, but only reapplies the balance impacts of the original event for these event types: adjustment, payment, writeoff, dispute, settlement, refund, charge, item transfer, cycle fold, cycle tax, reversal, and pre-rated.

If there is no balance impact associated with an event or if there is no rated quantity on the original event, the event is not rerated.

The input to PCM_OP_SUBSCRIPTION_RERATE_REBILL includes:

- The POID of the account to rerate.
- The time from which to start rerating. Events occurring from this start time are rerated.
- Flags that modify rerating behavior. See "Flags Used for Rerating".
- The session object. When PCM_OP_SUBSCRIPTION_RERATE_REBILL is invoked by the **pin_rerate** utility, the session object passed is the rerating audit object (**/event/control_rerate**). This object is used to generate rerating reports.

PCM_OP_SUBSCRIPTION_RERATE_REBILL does the following:

1. Closes the billing cycle, if due.
2. Retrieves all events for the account from the start date specified.
3. Determines the balance for each balance element in the account when rerating starts. This is the current balance minus the cumulative total of all balance impacts for the balance element from the rerating start time until the current time.
4. For each event:
   - If the effective time is passed in the PIN_FLD_START_T field, no balance impact is required and the event is not rerated.
   - Checks the **/data/ledger_report** object to determine whether the G/L for the event has been posted.

- Calls the PCM_OP_ACT_USAGE opcode to determine the difference between the original rated event and the rerated event.

> **Note:**
>
> PCM_OP_ACT_USAGE does not check credit limits when rerating events.

- Calls PCM_OP_ACT_USAGE again to apply the results of rerating to the account balance. If the event is sponsored, the balance impacts are applied to the sponsoring account.
- Updates the RERATE_OBJ field for the event being rerated with the POID of the new rerated event.

The account balance is updated after rerating each event so that each subsequent event is rerated based on the most current balance.

Some rerated events may be associated with a charge offer, discount offer, or balance element balance whose validity period starts on first usage. If the event that triggered the validity period is not the actual first usage event, PCM_OP_SUBSCRIPTION_RERATE_REBILL backs out the validity period of the charge offer, discount offer, or balance element balance and resets it based on the event time of the actual first-usage event.

When override pricing is passed to PCM_OP_SUBSCRIPTION_RERATE_REBILL, it creates a **/rerate_session/override_info** object to capture the override pricing information used during rerating.

Flags Used for Rerating

The PIN_FLD_FLAGS field specifies whether to rerate events in order based on the event creation time or the event end time. The end time is the default.

The PIN_FLD_RERATE_FLAGS field can take two flags:

- A flag that specifies back-out-only rerating. When this flag is set, the balance impacts of the events selected for rerating are backed out, but the events are not rerated.
- A flag that specifies selective rerating of specific events. When this flag is set, the account selection criteria is also applied to the account's events and only events that meet the selection criteria are rerated.

> **Note:**
>
> – Do not use selective rerating if your rating scheme includes credit limits or balance element-based tiers. These schemes require that all events related to an account are rerated to assure accurate rerating.
>
> – Do not use selective rerating if deferred taxation is used for taxing events during rerating.
>
> – Use caution when specifying selective rerating. Account balances can be impacted by different types of events and the order of the balance impacts is important to accurately apply discounts and consume balance elements. It is typically safer to rerate all of an account's events.

The selection criteria for selective rerating and back-out-only rerating must be set in the PIN_FLD_ARGS array field in the input flist.

Return Values for Rerating

If the CALC_ONLY flag is set or if PCM_OP_FLAG_READ_RESULT is passed in the input flist, PCM_OP_SUBSCRIPTION_RERATE_REBILL returns all rerating details in the PIN_FLD_RESULTS array without modifying the account balances. Otherwise, the account balances are updated and only the POID of the rerated event is returned.

If PCM_OP_SUBSCRIPTION_RERATE_REBILL attempts to rerate an event that is already being rerated, an error is returned.

How BRM Creates Rerate Jobs

A rerate job is a pair of **/job/rerate** and **/job/rerate** objects in the BRM database. There is a one-to-one relationship between **/job/rerate** objects and **/job_batch/rerate** objects and both are created in the same transaction when one or more accounts are selected for rerating.

You create rerate jobs manually by using the **pin_rerate** utility. BRM creates rerate jobs automatically by using the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode. This opcode is called to create rerate jobs by the following:

- The **pin_rerate** utility.
- BRM's automatic-rerating mechanism.

  BRM supports the automatic rerating of certain types of events. For more information, see "About Automatic Rerating".
- Your trigger-dependent rerating configuration.

  BRM enables you to rerate events automatically based on your own business requirements.
- Out-of-order rerating for pipeline rated events.

  BRM supports the automatic rerating of events that are rated out of order in the batch rating pipeline.

PCM_OP_RERATE_INSERT_RERATE_REQUEST receives the following information in the input flist:

- The POIDs of the accounts to be rerated (required).
- The time from which events must be rerated for the accounts (required).
- Accounts related to the accounts to be rerated (for example, an account that used the account's service before a line transfer) along with the start time for each account.

  > **Note:**
  >
  > If more than one account is included in the rerate job, the same rerate start time, selection criteria, and price overrides apply to all the accounts.

- Your rerate reason for trigger-dependent rerating (the default rerate reason passed in for BRM automatic rerating scenarios is **0**).
- Your selection criteria for trigger-dependent rerating to identify the events that must be rerated for the accounts.

- A list of charge offers, discount offers, or bundles to override the subscriber's existing pricing objects during rerating.

To create rerate jobs, PCM_OP_RERATE_INSERT_RERATE_REQUEST does the following:

1. Checks for duplicate rerate job requests.

   For more information, see "How BRM Handles Duplicate Rerate Jobs".

2. Calls PCM_OP_RERATE_CREATE_JOB to create the **/job/rerate** and **/job_bacth/rerate** objects.

How BRM Handles Duplicate Rerate Jobs

The PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode checks for existing jobs when a new rerate request is made; this opcode compares the data for each top-level account in the rerate request to the data in the existing jobs. To avoid duplicate rerate jobs, accounts are either eliminated from or updated in the rerate request or the existing jobs.

After the duplicate detection process is complete for each top-level account in the rerate request, PCM_OP_RERATE_INSERT_RERATE_REQUEST calls PCM_OP_RERATE_CREATE_JOB to create a rerate job with the resulting contents of the rerate request. PCM_OP_RERATE_CREATE_JOB then performs checks on the sub-accounts in the rerate request to check for duplicate jobs for line transfers before creating the rerate job.

Detecting Duplicate Rerate Requests

When the rerate reason code and rerate selection criteria for a rerate request match those of one or more rerate jobs, the request is considered a possible duplicate. When the rerate reason code and rerate selection criteria do not match, the rerate request is not a duplicate, and a new rerate job is created.

Avoiding Duplication of Rerate Jobs

To avoid duplicate rerate jobs for an account, PCM_OP_RERATE_INSERT_RERATE_REQUEST compares the following values for duplicate rerate job requests:

- The rerate start times. BRM always uses the earlier start time to ensure that all events are rated.
- The price overrides (if any). BRM uses the latest price overrides for an account because it is assumed to be the most updated.

When price overrides match

When the price overrides match, PCM_OP_RERATE_INSERT_RERATE_REQUEST does the following for each top-level account:

- If the rerate start time of the rerate request is later than or equal to the start time of the existing job, does the following:
  - If there is only this account in the rerate request, deletes the request. No rerate job is needed.
  - If there are many accounts in the new request, removes only that account from the new request.
- If the rerate start time of the rerate request is earlier than the start time of the existing job, does the following:

  - If the existing job contains only this account, updates the existing job to use the earlier start time and removes the account from the rerate request.
  - If the existing job contains other accounts, removes the account from the existing job, keeping the account in the rerate request.

When price overrides do not match

When the price overrides do not match, PCM_OP_RERATE_INSERT_RERATE_REQUEST does the following for each top-level account:

- If the rerate start time of the rerate request is the same or earlier than the start time of the existing job, does the following:
  - If there is only one account in the existing job, deletes the job.
  - If there are many accounts in the existing job, keeps the account in the rerate request so the new rerate job start time and price overrides are used. Removes the account from the existing job.
- If the rerate start time of the rerate request is later than the start time of the existing job, does the following:
  - If there is only one account in the existing job, deletes the job.
  - If there are many accounts in the existing job, keeps the account in the rerate request so the new price overrides are used. Updates the rerate request start time to use the existing job's start time. Removes the account from the existing job.

Table 63-4 summarizes how PCM_OP_RERATE_INSERT_RERATE_REQUEST handles duplicate rerate job requests:

Table 63-4 Request Handling by PCM_OP_RERATE_INSERT_RERATE_REQUEST

If the Start Time of the New Request Is:	And the Price Overrides:	Perform This Action:
Equal to or later than that of the existing job	Match	If there is only this account in the rerate request, delete the request. No rerate job is needed. If there are many accounts in the rerate request, remove only that account from the rerate request.
Earlier than that of the existing job	Match	If there is only one account in the existing job: • Update the existing job to use the earlier start time. • Remove the account from the rerate request. If there are many accounts in the existing job: • Keep the account in the rerate request. • Remove the account from the existing job.
Later than that of the existing job	Do not match	If there is only one account in the existing job, delete the job. If there are many accounts in the existing job: • Keep the account in the rerate request so the new price overrides are used. • Update the rerate request job start time to use the existing job's start time. • Remove the account from the existing job.

Table 63-4 (Cont.) Request Handling by PCM_OP_RERATE_INSERT_RERATE_REQUEST

If the Start Time of the New Request Is:	And the Price Overrides:	Perform This Action:
Equal to or earlier than that of the existing job	Do not match	If there is only one account in the existing job, delete the existing job. If there are many accounts in the existing job: • Keep the account in the rerate request so the rerate request start time and price overrides are used. • Remove the account from the existing job.

For more information on how BRM creates rerate jobs, see "How BRM Creates Rerate Jobs".

Rerating Cycle Fees

To rerate cycle fees, use the PCM_OP_SUBSCRIPTION_RATE_CHANGE opcode. This opcode uses the event notification mechanism to trigger the creation of rerating requests when there is a cycle-forward or cycle-forward-arrears event rate change in the middle of the current cycle.

> **Note:**
>
> Rerating is not triggered for cycle-arrears rate changes or rate changes in future cycles.

When you run the **pin_rate_change** utility after a charge pricing change, the utility calls this opcode and provides details about the accounts and charge offers affected by the rate change.

This opcode returns a notification event of type **/event/notification/rate_change** for each account picked up by the **pin_rate_change** utility. Depending on how automatic rerating is configured, the notification event triggers the creation of rerating requests.

64

Configuring Comprehensive Rerating

This chapter describes how to configure your Oracle Communications Billing and Revenue Management (BRM) system for comprehensive rerating using **pin_rerate**.

> **Note:**
>
> This documentation does not apply to rerating only pipeline-batch-rated events by using a batch rerating pipeline.

About Configuring Comprehensive Rerating

How you configure rerating depends on whether you rerate real-time-rated and pipeline-batch-rated events concurrently or, if you do not use Pipeline Manager for batch rating, rerate only real-time-rated events. See one of the following sections:

- Configuring Concurrent Rerating of Pipeline-Rated and Real-Time-Rated Events
- Configuring Rerating When You Do Not Use a Batch Rating Pipeline

You perform the following configurations for both concurrent rerating and real-time-event only rerating:

- Specifying Whether the Batch Rating Pipeline Is Enabled
- (Optional) "Setting the Rerating Event Cache Size (Fetch Size)"
- (Optional) "Configuring the Number of Accounts Per Job and Number of Jobs per Transaction"

You can also configure BRM to automatically create rerate jobs when the following events occur:

- When rates are changed.
- When rollover corrections are made due to delayed events.
- When certain events, such as purchase and cancellation events, are backdated. See "About Automatic Rerating of Backdated Events".
- When pipeline batch-rated events are rated out of order. See "About Automatic Rerating of Out-of-Order Events".
- When events trigger rerating based on your custom rerating requirements. See "About Trigger-Dependent Rerating".

Configuring Concurrent Rerating of Pipeline-Rated and Real-Time-Rated Events

> **Note:**
>
> Before you can configure concurrent rerating, you must have installed and configured Pipeline Manager and the Account Synchronization Data Manager (DM).

To configure concurrent rerating of both pipeline-rated and real-time-rated events, perform these tasks:

- Configure a real-time rerating pipeline. See "Configuring a Real-Time Rerating Pipeline".
- Configure **pin_rerate** and the batch rating pipeline. See "Configuring the Batch Rating Pipeline and pin_rerate to Synchronize Processing".
- (Optional) Configure the rerating event cache size. See "Setting the Rerating Event Cache Size (Fetch Size)".
- (Optional) Configure parameters for creating rerate jobs. See "Configuring the Number of Accounts Per Job and Number of Jobs per Transaction".

Configuring a Real-Time Rerating Pipeline

A real-time rerating pipeline is configured similarly to a batch rerating pipeline, but with fewer preprocessing and post-rating modules. This is because most of the enrichment data is provided in the input flist received from the CM.

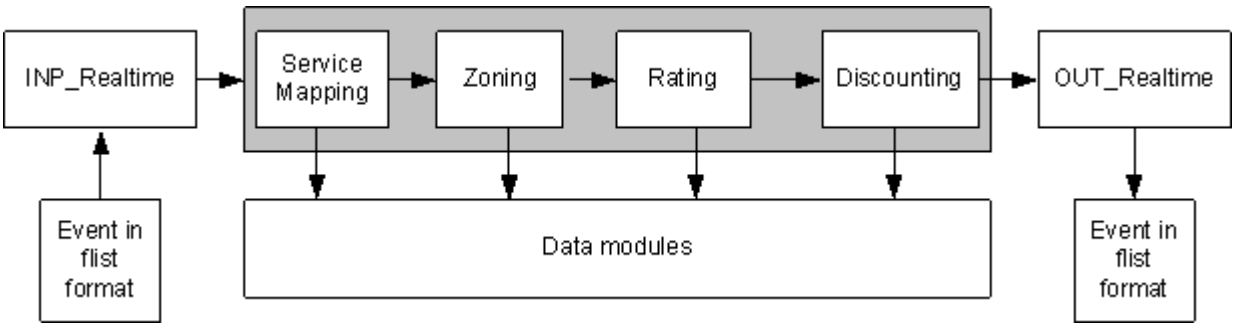
You typically create a separate instance of Pipeline Manager for real-time rerating. The default registry file is *BRM_home***/conf/wirelessRealtime.reg**.

To configure a real-time rerating pipeline, perform the following tasks:

- (Optional) "Configuring Multiple Real-Time Rerating Pipelines"
- Configuring the Real-Time Rerating Data Pool
- Configuring the Modules in the Real-Time Rerating Pipeline
- Adding Real-Time Rerating Pipeline Data to the IFW_PIPELINE Table

Figure 64-1 shows the real-time rerating pipeline architecture:

Figure 64-1 Real-Time Rerating Pipeline Architecture



Configuring Multiple Real-Time Rerating Pipelines

To improve performance or scalability, you configure multiple instances of real-time rerating pipelines. For example, you can increase performance by configuring multiple pipelines to process rerate requests in parallel or by configuring multiple real-time rerating pipelines to process different rerate requests: for example, by configuring one pipeline that rerates only GPRS events and another that rerates only GSM events.

Configuring the Real-Time Rerating Data Pool

Configure the following data modules:

- DAT_Zone
- Rateplan
- DAT_Calendar
- DAT_TimeModel
- DAT_PriceModel
- Dayrate
- DAT_Currency
- DAT_USC_Map

Configuring the Modules in the Real-Time Rerating Pipeline

Configure the following modules in the real-time rerating pipelines:

- Configure the INP_Realtime input module. Specify the flist-to-EDR mapping file used for rerating in the **OpcodeMapping** registry entry. For example:

  OpcodeMapping = ./formatDesc/Formats/Realtime/rate_event.xml

- Configure the NET_EM module:
  - Configure NET_EM to manage data between the CM and Pipeline Manager and configure the CM to send rerate request to the NET_EM module.
  - Configure NET_EM to route rerate requests. See "Configuring NET_EM to Route Rerate Requests Based on the Event Field Value".
- Configure the output module to add any customizations to the output flist.
- Configure these function modules:
  - FCT_ServiceCodeMap
  - FCT_CustomerRating
  - FCT_PreRating
  - FCT_IRules
  - FCT_USC_Map
  - FCT_RSC_Map
  - FCT_MainRating
  - FCT_Dayrate
  - FCT_RateAdjust

- FCT_Rounding
- FCT_DiscountAnalysis
- FCT_Discount

Configuring NET_EM to Route Rerate Requests Based on the Event Field Value

To configure NET_EM to route rerate requests to multiple real-time rerating pipelines based on the type of event, you set the **FieldName** and **FieldValue** NET_EM module registry entries.

> **Note:**
>
> If you use event routing based on the event field value, ensure that the input events contain the expected field name and field values specified in the NET_EM module registry. Otherwise, NET_EM will not be able to route the events.

By using the "**.**" notation, you can specify a field at any level in the input event flist to be used to route the event. For example, this substruct and field:

```
PIN_FLD_EVENT
     PIN_FLD_POID
```

is represented like this:

```
PIN_FLD_EVENT.PIN_FLD_POID
```

In the NET_EM registry below, if PIN_FLD_EVENT.PIN_FLD_POID is a GSM event, the rerate request is routed to any one of the two instances of the GSM rerating pipeline (RealtimeReratingGSM). If the event is a GPRS event, the rerate request is routed to any one of the two instances of the GPRS rerating pipeline (RealtimeReratingGPRS).

```
DataPool
{
     RealtimePipeline
     {
          ModuleName = NET_EM
          Module
          {
               ThreadPool
               {
                   Port = 14579
                   Threads = 4
               }

               ReratingOpcode
               {
                   OpcodeName = PCM_OP_RATE_PIPELINE_EVENT
                   FieldName = PIN_FLD_EVENT.PIN_FLD_POID
                   GSMEvent
                   {
                     FieldValue = /event/delayed/session/telco/gsm
                     PipelineName = RealtimeReratingGSM
                     NumberOfRTPipelines = 2
                   }
                   GPRSEvent
                   {
                     FieldValue = /event/delayed/session/gprs
                     PipelineName = RealtimeReratingGPRS
```

```
                    NumberOfRTPipelines = 2
                }
            }

        }
    }
}
```

Adding Real-Time Rerating Pipeline Data to the IFW_PIPELINE Table

Pipeline Manager stores information about pipelines in the IFW_PIPELINE table. The pipelines that are preconfigured in the **Pipelines** section of the default registry file (*pipeline_home***/conf/wirelessRealtime.reg**) are inserted into the IFW_PIPELINE table during Pipeline Manager installation.

> **Note:**
>
> - If you are *not* using the default registry file, and you have configured new real-time rerating pipelines, you must manually insert the pipelines into the IFW_PIPELINE table by using SQL commands. Otherwise, you will receive an error at system startup.
> - If you *are* using the default registry file and have changed the default pipeline names or you have configured additional pipelines, you must manually insert the new pipelines into the IFW_PIPELINE table.

The following example shows SQL commands to insert RealtimeReratingGSM and RealtimeReratingGPRS pipelines into the IFW_PIPELINE table:

```
% sqlplus pin@databaseAlias
Enter password: password

SQL>INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC ) VALUES
( 'RealtimeReratingGSM', 'GSM Realtime Rerating Pipeline', 'ALL_RATE');

SQL>INSERT INTO IFW_PIPELINE ( PIPELINE, NAME, EDRC_DESC ) VALUES
( 'RealtimeReratingGPRS', 'GPRS Realtime Rerating Pipeline', 'ALL_RATE');

SQL>commit;
```

Configuring the Batch Rating Pipeline and pin_rerate to Synchronize Processing

To enable **pin_rerate** and the batch rating pipeline to synchronize processes, perform the following tasks:

- Ensure that **pin_rerate** can communicate with a batch pipeline by setting the batch rating pipeline business parameter to **enabled**. See "Specifying Whether the Batch Rating Pipeline Is Enabled ".
- Configure rerating business events and event notification. See "Configuring Rerating Business Events and Event Notification".
- Configure Pipeline Manager to send and receive events. See "Configuring Pipeline Manager to Handle Business Events from pin_rerate".

- Configure **pin_rerate** to send and receive events. See "Configuring pin_rerate to Receive Acknowledgment Events from Pipeline Manager".
- Configure standard recycling to enable Pipeline Manager to suspend and recycle EDRs. See "Configuring Standard Recycling".

Configuring Rerating Business Events and Event Notification

The **pin_rerate** utility and the batch rating pipeline synchronize processing by passing business events by way of the Account Synchronization DM. To generate business events, **pin_rerate** uses event notification.

The business events and notification events used for rerating are defined in configuration files provided with the Account Synchronization DM:

- The notification events used for rerating are specified by the following entries in the Account Synchronization DM notification list (*BRM_home***/sys/data/config/pin_notify_ifw_sync**):

```
1301    0    /event/notification/rerating/start
1301    0    /event/notification/rerating/end
1301    0    /event/notification/rerating/PrepareToRerate
1301    0    /event/notification/rerating/ReratingCompleted
```

- The business events used for rerating are specified in the Account Synchronization DM EAI payload configuration file (*BRM_home***/sys/eai_js/payloadconfig_ifw_sync.xml**).

You configure these files when you install and configure the Account Synchronization DM.

Configuring Pipeline Manager to Handle Business Events from pin_rerate

To enable **pin_rerate** and Pipeline Manager to send and receive business events through the Account Synchronization DM, perform these tasks:

- Creating the Acknowledgment Queue
- Configuring Access to Queues in a Multischema System
- Configuring the DAT_Listener Module

Creating the Acknowledgment Queue

The Account Synchronization DM uses database queues to send business events to Pipeline Manager. A default database queue is created for this purpose when you install the Account Synchronization DM.

You must create an additional database queue for Pipeline Manager to post acknowledgment events that are dequeued and processed by **pin_rerate**.

> **Note:**
>
> Only one business event queue and one acknowledgment queue is required for each instance of Pipeline Manager in your system. You do not need to create additional acknowledgment queues if one has already been created.

To create an acknowledgment queue, use the **pin_ifw_sync_oracle.pl** utility. For example:

```
pin_ifw_sync_oracle.pl create -q ACK_QUEUE -t ack_queue_t -l /@database
```

Configuring Access to Queues in a Multischema System

You can run BRM and Pipeline Manager in a multischema environment. For example, you might have a schema for BRM that uses the **PIN** logon, a schema for Account Synchronization queue that uses the **PINQ** logon, and a schema for the instance of Pipeline Manager that uses the **INTEGRATE** logon.

In such cases, you must create a global synonym for the Account Synchronization acknowledgment queue. This enables the user of the BRM schema to access the acknowledgment queue in the Account Synchronization schema.

1. Using SQL*Plus, log in to your database as the SYSTEM user:

   ```
   % sqlplus system@DatabaseAlias
   Enter password: password
   ```

2. Create a global user synonym:

   ```
   SQL> CREATE PUBLIC SYNONYM ACCT_SYNC for PINQ.ACCT_SYNC
   ```

   where *PINQ* is the login for the schema that includes the Account Synchronization stored procedures and acknowledgment queue.

   > **Note:**
   >
   > The rerating stored procedures run in the PIN schema (the BRM schema).

3. Type **exit** to exit SQL*Plus.
4. In the **pin_rerate** configuration file (*BRM_home***/app/pin_rerate/pin.conf**), add the global synonym:

   ```
   - pin_rerate ack_queue PINQ.ACK_QUEUE
   ```

Configuring the DAT_Listener Module

Pipeline Manager responds to business events sent by **pin_rerate** by posting acknowledgment events. You must configure the DAT_Listener module to post acknowledgment events to the Account Synchronization acknowledgment queue. Set the **AckQueueName** entry to specify the name of the acknowledgment queue.

Sample DAT_Listener registry:

```
Listener
{
     ModuleName = DAT_Listener
     Module
     {
          InfranetConnection = ifw.DataPool.LoginQueue
          QueueName = QUEPIN117
          AckQueueName = ACK_QUEUE
          MQAckServerName = ACK_QUEUE_SERVER
          LogEvents = TRUE
     }
}
```

Configuring pin_rerate to Receive Acknowledgment Events from Pipeline Manager

To enable **pin_rerate** to receive acknowledgment events from Pipeline Manager, perform the following tasks:

- Loading the Stored Procedure for Rerating
- Configuring pin_rerate to Dequeue Events from the Acknowledgment Queue

Loading the Stored Procedure for Rerating

The **pin_rerate** utility uses stored procedures to dequeue acknowledgment events from the database queue and to purge rerate jobs.

> **Note:**
>
> - Before loading the **pin_rerate** stored procedures, you must have Account Synchronization installed and configured.
> - On Oracle systems, you manually load the rerating stored procedures **job_rerate_procedures_pkb.plb** and **job_rerate_procedures_oracle.plb** into your BRM database:

To load the stored procedure:

1. Connect to your database using SQL*Plus:

   ```
   % sqlplus user@databaseAlias
   Enter password: password
   ```

   where *user* and *password* are your user name and password, and *databaseAlias* is the service name for the BRM database.

2. At the SQL*Plus prompt, enter the following commands:

   ```
   SQL> @BRM_home/sys/dm_oracle/data/job_rerate_procedures_pkg_oracle.plb
   SQL> @BRM_home/sys/dm_oracle/data/job_rerate_procedures_oracle.plb
   ```

   where *BRM_home* is the directory in which BRM is installed.

Configuring pin_rerate to Dequeue Events from the Acknowledgment Queue

Configure the **pin_rerate** utility to dequeue events from the acknowledgment queue that you created in "Creating the Acknowledgment Queue".

Add the following entry in the **pin_rerate** configuration file (*BRM_home***/app/pin_rerate/pin.conf**):

```
- pin_rerate ack_queue ACK_QUEUE
```

where ACK_QUEUE is the name of the acknowledgment queue.

Configuring Standard Recycling

BRM rerating uses standard recycling for two purposes:

- To load EDRs suspended during rerating into the BRM database.
- To retrieve suspended EDRs from the BRM database and recycle them when rerating is complete.

Configuring Rerating When You Do Not Use a Batch Rating Pipeline

If you do not use Pipeline Manager for batch rating, perform the following tasks to configure **pin_rerate** to rerate only real-time-rated events:

- Ensure that **pin_rerate** does not attempt to communicate with a batch rating pipeline by setting the batch rating pipeline business parameter to **disabled**. See "Specifying Whether the Batch Rating Pipeline Is Enabled ".
- (Optional) Configure the rerating event cache size. See "Setting the Rerating Event Cache Size (Fetch Size)".
- (Optional) Configure parameters for creating rerate jobs. See "Configuring the Number of Accounts Per Job and Number of Jobs per Transaction".

Specifying Whether the Batch Rating Pipeline Is Enabled

When the batch pipeline is enabled, **pin_rerate** generates notification events to synchronize processing with the batch pipeline by way of the Account Synchronization DM. When the batch pipeline is disabled, **pin_rerate** does not generate notification events because synchronization is not needed.

pin_rerate checks a field in the **rerate** instance of the **/config/business_params** object, to determine if the batch rating pipeline is enabled or disabled:

- To rerate both real-time-rated and pipeline-rated events concurrently, the batch rating pipeline entry must be set to **enabled**.
- To rerate only real-time-rated events, the batch rating pipeline entry must be set to **disabled**.

The default is **enabled**.

You modify the **/config/business_params** object by using the **pin_bus_params** utility.

To specify whether the batch pipeline is enabled or disabled:

1. Use this command to create an editable XML file from the **rerate** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

   This command creates the XML file named **bus_params_rerate.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:

   ```
   <BatchRatingPipeline>enabled</BatchRatingPipeline>
   ```

3. Set the entry as needed:
   - **enabled**: To rerate both real-time-rated and pipeline-rated events concurrently.

- **disabled:** To rerate only real-time-rated events when you do not use Pipeline Manager for batch rating.

> **Note:**
>
> BRM uses the XML in this file to overwrite the existing **rerate** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM billing configuration.

4. Save the file and change the file name from **bus_params_rerate.xml.out** to **bus_params_rerate.xml**.
5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_rerate.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility.
6. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
7. Stop and restart the Connection Manager (CM).
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

Setting the Rerating Event Cache Size (Fetch Size)

By default, BRM rerating caches 10,000 events in system memory for processing. Depending on your system memory, you can set the **event_fetch_size** in the Connection Manager's configuration file to specify the number of events retrieved from the database and cached in the system memory for processing.

To set the event cache size:

1. Open the Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf)** using a text editor**.**
2. Uncomment the **- fm_subscription event_fetch_size** entry.
3. Edit the **event_fetch_size** value. The default is 10000.

   ```
   - fm_subscription event_fetch_size 10000
   ```
4. Save the file.
5. Stop and restart the CM.

Configuring the Number of Accounts Per Job and Number of Jobs per Transaction

By default, BRM assigns 10 accounts to each rerate job and creates 2 rerate jobs per transaction. For example, if the total number of accounts to rerate is 10,000, BRM creates 1000 rerate jobs. It creates the rerate jobs in 500 separate transactions: 2 jobs in each transaction.

To change the default number of accounts per job and the number of rerate jobs created per transaction, perform the following tasks:

1. Open the **pin_rerate** configuration file (*BRM_home***/apps/pin_rerate/pin.conf**).
2. Set the number of accounts assigned to each rerate job by adding the following line:

   ```
   - pin_rerate per_job accounts_per_job
   ```

   where *accounts_per_job* is the number of accounts to assign to each job.
3. Set the number of jobs created per transaction by adding the following line:

   ```
   - pin_rerate per_transaction jobs_per_transaction
   ```

   where *jobs_per_transaction* is the number of jobs to create in each transaction.
4. Save and close the file.

> **Note:**
>
> Setting the **pin_rerate per_job** entry to a small number, for example 1, will result in many rerate jobs being created. Too many rerate jobs can affect your system's performance due to the rerate steps performed for each rerate job. Processing multiple accounts in one rerate job reduces the total number of rerate steps performed compared to processing those same accounts in multiple rerate jobs.

Configuring Rerating to Reset First-Usage Validity Periods

Perform this task if your product offerings include charge offers, discount offers, or balance impacts that become valid on first usage (when customers use the charge offers and discount offers or consume the balance element balance for the first time).

When rerated events are associated with charge offers, discount offers, or balance element balances that start on first usage, rerating resets their validity periods, if necessary. For example, if the first event rated, which initiated a charge offer's validity period, was not actually the first event to use the charge offer's service, rerating corrects the order of the events and resets the validity period based on the actual first-usage event.

To configure rerating for first-usage validity, perform the tasks in the following sections:

- Configuring the Real-Time Rerating Pipeline to Set Charge Offer Validity Periods.
- Configuring Event Notification for Rerating Cycle Events Triggered by First Usage.

Configuring the Real-Time Rerating Pipeline to Set Charge Offer Validity Periods

If your charge offers are configured to start when they are first used, configure the ISC_FirstProductRealtime iScript in the real-time rerating pipeline.

When charge offers start on first usage, the real-time rerating pipeline adds the account's first-usage charge offer and discount offer information to the EDR. ISC_FirstProductRealtime sets the validity period in the BRM database for charge offers that were used to rate the event and that start on first usage. This triggers any purchase and cycle fees.

Configuring Event Notification for Rerating Cycle Events Triggered by First Usage

To ensure that BRM can rerate cycle events that are triggered because of first usage, you must add cycle-event notification events to your event notification list (the default notification list is *BRM_home*/**sys/data/config/pin_notify**). BRM provides a list of default cycle-event notification events in the **pin_notify_rerate** file. Use this file to update your **pin_notify** file.

To configure event notification for rerating cycle events triggered by first usage:

1. Open the **pin_notify** and **pin_notify_rerate** files in *BRM_home*/**sys/data/config**.
2. Copy the entries in the **pin_notify_rerate** file to the end of the **pin_notify** file.
3. Save and close the **pin_notify** file and close the **pin_notify_rerate** file.
4. Load the contents of the **pin_notify** file into the **/config/notify** object in the BRM database by running the **load_pin_notify** utility.

Configuring Rerating for Accounts Associated With Subscription Service Transfer

When a subscription service is transferred from one subscriber account to another, rerating the original account can affect the subscription service balances transferred to the new account. However, by default, the rerating process only selects accounts for rerating based on your specified search criteria. This means that, when an account selected for rerating is associated with a subscription service transfer, the other accounts to which the subscription service was transferred are not selected for rerating.

To rerate all accounts associated with a subscription service transfer, you configure the **LineManagement** business configuration parameter. When this parameter is enabled, the rerating process searches for any subscription service transfers for the account that is rerated, and adds the accounts to which the subscription service was transferred during the rerating time specified to the rerate request.

For example:

1. Subscription service X is originally owned by Account A and transferred to Account B on June 15.
2. On June 20, the service is transferred again from Account B to Account C.
3. On July 10, Account A is selected for rerating and the rerating time specified is June 1. Rerating also selects Accounts B and C because the subscription service transfer to Account B and C occurred after June 1. Accounts A, B, and C are grouped together to form a single rerate request.
4. Account A is rerated from June 1 to July 10, Account B is rerated from June 15 to July 10, and Account C is rerated from June 20 to July 10. Rerating selects the events for accounts A, B, and C, and rerates the events in chronological order, resulting in correct subscription service balance updates.

> **Note:**
>
> If rerating fails for any one of the accounts in the rerate request, rerating fails for all the accounts in the rerate request.

During rerating, each account is locked only for as long as it takes to rerate its events. However, in case of a subscription service transfer, related accounts are locked for the duration of rerating events associated with all of those accounts. In the example above, accounts A, B, and C remain locked until rerating of all three accounts is complete.

By default, the **LineManagement** parameter is disabled and cached at CM startup. To change the default value, you modify the **rerate** instance of the **/config/business_params** object by using the **pin_bus_params** utility.

To enable rerating of accounts associated with a subscription service transfer:

1. Run this command to create an editable XML file from the **rerate** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRerate bus_params_rerate.xml
   ```

   This command creates the XML file named **bus_params_rerate.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.
2. Search the XML file for following line:

   ```
   <LineManagement>0</LineManagement>
   ```
3. Change **0** to **1**.

   > **Note:**
   >
   > BRM uses the XML in this file to overwrite the existing **rerate** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of BRM's billing configuration.
4. Save the file and change the file name from **bus_params_rerate.xml.out** to **bus_params_rerate.xml**.
5. Go to the *BRM_home***/sys/data/config** directory.
6. Run the command:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_rerate.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_rerate.xml** resides.
7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
8. Stop and restart the Connection Manager (CM).
9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About Automatic Rerating of Backdated Events

> **Note:**
>
> Automatic rerating *does not* mean that rerating occurs immediately when an event is backdated. Automatic rerating means that rerate jobs (**/job/rerate** and **/job_rerate/rerate** objects) are automatically created.

When backdated events occur, BRM uses event notification to automatically create rerate jobs to rerate the backdated events. The accounts in the rerate jobs are rerated the next time you run **pin_rerate** to process rerate jobs; you do not need to specify the accounts and events in the **pin_rerate** command line.

BRM supports automatic rerating of backdated events in the following cases:

- When purchase or cancellation of a charge offer, discount offer, or bundle is backdated.
- When adjustment to a noncurrency balance element is backdated.
- When an extended rating attribute (ERA) modification is backdated.

Table 64-1 shows the entries in the CM configuration file used to create rerate jobs for backdated events:

Table 64-1 Configuration File Entries for Rerate Jobs

Entry	Description
backdate_trigger_auto_rerate	Specifies whether automatic rerating is enabled.
backdate_window	Specifies the minimum time difference needed between the current time and the backdated event time for triggering automatic rerating.
num_billing_cycles	Specifies the maximum number of billing cycles allowed between the current time and the backdated event time of a backdated operation.

> **Note:**
>
> All of these conditions must be met to trigger automatic rerating. Otherwise, the backdated event is not rerated.

By default, BRM creates rerate jobs for backdated events when the following conditions are met:

- Automatic rerating is enabled.
- The backdated event time is at least one hour earlier than the current time.
- The backdated event date is not older than one billing cycle.

You can change the default backdated time and date thresholds in the CM configuration file. See "Configuring Automatic Rerating of Backdated Events".

You can backdate beyond the number of billing cycles specified in the **num_billing_cycles** entry without requesting to automatically rerate. For more information, see "Backdating beyond Configured Billing Cycles without Automatic Rerating Request".

About Backdated Bundle, Charge Offer, and Discount Offer Purchase

Automatic rerating of backdated bundle, charge offer, and discount offer purchases is triggered when the difference between the *current time* and the backdated event *purchase, cycle,* or the *usage start time* is greater than the time specified by the **backdate_window** parameter.

BRM then validates that the backdated purchase occurred within the number of billing cycles specified by the **num_billing_cycles** parameter. If the validation is successful, the account is automatically rerated the next time you run **pin_rerate**.

For example, a subscriber's billing day is on the first day of the month. The subscriber purchases a charge offer on July 19 at 5:00 a.m. The CSR backdates the purchase start date to July 15. If the **backdate_window** is set to **1** hour and **num_billing_cycles** is set to **1** cycle, BRM validates that the purchase start time is on or before 4:00 a.m., July 19, and that the purchase start date is not earlier than June 1 (one billing cycle). Because both conditions are met and automatic rerating is enabled, when **pin_rerate** is run with **-rerate** parameter on July 20, BRM automatically rerates all the account's events that occurred after midnight of July 14 to July 20.

About Backdated Bundle, Charge Offer, and Discount Offer Cancellation

Automatic rerating of backdated bundle, charge offer, and discount offer cancellation is triggered when the difference between the *current time* and the *purchase*, *cycle*, or the *usage end time* is greater than the time specified by the **backdate_window** parameter.

BRM then validates that the backdated cancel occurred within the number of billing cycles specified by the **num_billing_cycles** parameter. If the validation is successful, the account is automatically rerated the next time you run **pin_rerate**.

The following example demonstrates how fees are prorated and charges are reversed when a backdate charge offer cancellation occurs.

In this example, the backdate parameters are as follows:

- **backdate_window** is 2 hours.
- **num_billing_cycles** is 2 cycles.

The subscriber's charge offer includes the following:

- IP charge offer purchase fee: $10
- IP charge offer monthly cycle forward fee: $20 (with proration enabled)
- IP charge offer cancellation fee: $50
- IP usage fee: $1 per minute
- Email charge offer monthly cycle forward fee: $8

1. On January 1, the subscriber purchases the charge offer.

   Account balance = $10 IP charge offer purchase fee + $20 IP monthly cycle forward fee + $8 email cycle forward fee = $38

2. On January 20, the subscriber generates a usage of $10 for the IP service.

   Account balance = $38 + $10 usage = $48

3. On February 1, when billing is run:

   Account balance = $48 + $20 IP monthly cycle forward fee (for February) + $8 email monthly cycle forward fee (for February) = $76

4. On February 10 at 5:00 p.m., the subscriber cancels the IP charge offer. The CSR backdates the charge offer cancellation to January 10. BRM automatically creates a rerate job and the backdate cancel event is rerated by running **pin_rerate**. After backdate cancellation of the IP Charge Offer:

   - A $50 IP charge offer cancellation fee is applied.
   - The IP charge offer monthly cycle forward fee for January is prorated and charged for 10 days only.
   - The $10 IP service usage fee is reversed.
   - The IP charge offer monthly cycle forward fee for February is reversed.

   Account balance = $10 IP purchase fee + $20 IP monthly cycle forward fee (for January) - $14.19 prorated IP monthly cycle forward fee refund + $8 email monthly cycle forward fee (for January) + $50 IP charge offer cancellation fee + $8 email cycle forward fee (for February) = $81.81

5. On March 1, when billing is run:

   Account balance = $81.81 + $8 email cycle forward fee (for March) = $89.81

BRM validates that the purchase end time is on or before 3:00 p.m., February 10, and that the purchase end date is not earlier than December 1 (two billing cycles). When both conditions are met and automatic rerating is enabled, when **pin_rerate** is run, BRM automatically rerates all the account's events that occurred after midnight January 10 to February 10.

About Backdated Adjustment of Noncurrency Balance Elements

When you backdate an adjustment of a noncurrency balance element and automatic rerating is enabled, BRM automatically rerates all the associated events the next time you run **pin_rerate** with the **-rerate** parameter.

> **Note:**
>
> - The adjustment event is rerated only if the adjustment is for a noncurrency balance element.
> - The adjustment must be backdated so that it can be used to rerate the events.

In the following example, the billing date is the first day of the month:

On February 15, the free minutes balance element is adjusted from 100 to 500 and the adjustment is backdated to be effective January 15. The **backdate_window** is 24 hours and **num_billing_cycles** is 1 cycle. When **pin_rerate** is run on February 20 with the **-rerate** parameter, BRM automatically rerates all the relevant events from January 15 to February 20 and applies the balance changes to the current billing cycle. It creates adjustment events for the events that occurred from January 15 to February 1 and shadow events for the events that occurred from February 1 to February 20. Billing events that previously occurred on February 1, such as billing-time discounts, rollovers, and folds, are recalculated based on the new adjustment and are reapplied.

About Backdated ERA Modifications

BRM automatically rerates backdated ERA modifications when the validity start time or end time is backdated and automatic rerating is enabled.

For example, a subscriber changes their service-level agreement from Silver to Gold on July 12. The CSR backdates the change to July 1 to let the subscriber apply the Gold-level benefits to all usage for July. The next time you run **pin_rerate**, BRM rerates all the relevant events for the account that occurred between July 1 and the current time.

Configuring Automatic Rerating of Backdated Events

To configure automatic rerating of backdated events, you perform the following tasks:

- Setting Thresholds That Trigger Automatic Rerating
- Configuring Event Notification for Rerating Backdated Events

Setting Thresholds That Trigger Automatic Rerating

BRM automatically rerates certain backdated events based on the default CM configuration settings. See "About Automatic Rerating of Backdated Events".

To change the default settings:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**) in a text editor.
2. To turn automatic rerating on or off, set the **backdate_trigger_auto_rerate** entry: **1** = enabled; **0** = disabled.

   ```
   - fm_subs backdate_trigger_auto_rerate 1
   ```

3. To specify the minimum time difference necessary to trigger automatic rerating, set the **backdate_window** entry. This is the amount of time in seconds between the current time and the backdated time. The default is 3600 seconds.

   ```
   - fm_subs backdate_window 3600
   ```

4. To specify the maximum number of billing cycles allowed between the current time and the backdated event date for triggering automatic rerating, Set the **num_billing_cycles** entry. The default is 1 billing cycle.

   ```
   - fm_subs num_billing_cycles 1
   ```

5. Save and close the file.
6. Stop and restart the CM.

Configuring Event Notification for Rerating Backdated Events

When a backdated event occurs, BRM rerating uses event notification to trigger automatic rerating of the event.

Although any subclass of the **/event** class can be used to trigger event notification, BRM rerating generates the nonpersistent **/event/notification/auto_rerate** event specifically to use for event notification.

By default, when this event occurs, BRM creates a rerate job.

Before you can use BRM rerating, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them.
2. Ensure that the merged file includes the following information from the *BRM_home*/**sys/data/config/pin_notify** file:

```
# Rerating related event notification
3787    0    /event/notification/auto_rerate
```

3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list.
4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger.
5. Load your final event notification list into the BRM database.

Backdating beyond Configured Billing Cycles without Automatic Rerating Request

You can backdate an event beyond the configured number of billing cycles without requesting to automatically rerate such an event.

To do so:

1. Ensure that automatic rerating is enabled for backdated events. If necessary, enable it by setting the **backdate_trigger_auto_rerate** entry in CM configuration file to **1**.
2. Enable the **AllowBackdateNoRerate** business parameter in the **/config/business_params** object by using the **pin_bus_params** utility.
3. After you enable the **AllowBackdateNoRerate** business parameter, you must manually rerate any events backdated *beyond* the number of billing cycles specified in the **num_billing_cycles** entry.

Enabling the AllowBackdateNoRerate Business Parameter

To set the **AllowBackdateNoRerate** business parameter to **enabled**:

1. Go to *BRM_home*/**sys/data/config** directory.
2. Create an editable XML file of the subscription instance from the **/config/business_params** object by using the following command:

```
pin_bus_params -r -c "Subscription" bus_params_subscription.xml
```

   BRM places the XML file named **bus_params_subscription.xml.out** in your working directory. To place this file in a different directory, specify the full path as part of the file name.
3. Locate the **AllowBackdateNoRerate** entry in the **bus_params_subscription.xml.out** file.
4. Set the value of **AllowBackdateNoRerate** to **enabled**, if necessary.

```
<AllowBackdateNoRerate>enabled</AllowBackdateNoRerate>
```

> **Caution:**
>
> BRM uses the XML in this file to overwrite the existing subscription instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM subscription configuration.

5. Save this updated file as **bus_params_subscription.xml**.
6. Load the modified XML file containing the business parameters for billing into the appropriate **/config/business_params** object in the BRM database.

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility.
7. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct. BRM stores one of the following values for **AllowBackdateNoRerate**:
   - **0** to indicate **disabled**.
   - **1** to indicate **enabled**.
8. Stop and restart Connection Manager.
9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

About Automatic Rerating of Out-of-Order Events

Events are processed by Pipeline Manager in the order that call details records (CDRs) are received. If the CDRs are sent out of order to Pipeline Manager, the events are processed out of order as well. Usually, this is not a problem; however, correct rating sometimes depends on rating events in chronological order (for example, when usage counters are used).

You can configure BRM to detect events that must be rated in chronological order and rerate them. To use out-of-order rerating, you define the criteria for when an out-of-order event must be rerated. When an event is rated, the FCT_EventOrder module uses the criteria and the event timestamps to determine if the event needs to be rerated.

About Detecting Out-of-Order Events

To enable out-of-order rerating, you configure criteria BRM uses to detect when events qualify for out-of-order rerating. Events must be rated in chronological order only when balance elements from the same balance group are affected, so the balance group is assumed as part of the detection criteria. If an account owns more than one service instance, each using a different balance group, out-of-order detection is applied to only the balance group associated with the event (the criteria name and balance group combination that is stored in the **/profile/event_ordering** object).

Pipeline Manager loads your detection criteria configuration at startup as follows:

- The DAT_PortalConfig module retrieves data from the **/config/event_order_criteria**object and loads it into its memory. This data specifies the criteria for determining if an event qualifies for out-of-order detection. The criteria is based on:
  - Service types
  - Event types

- Charge offers
- Discounts
- A combination of service types, event types, charge offers, and discounts

For more information, see "About Out-of-Order Rerating Criteria".

> **Note:**
>
> This feature supports branding. You can have a different configuration for each brand.

- The DAT_AccountBatch module retrieves data from the **/profile/event_ordering** profile object and loads it into its memory. This data includes the timestamp for the last time an event was processed for a particular billing cycle with the criteria specified in the **/config/event_order_criteria** object and the balance group determined when an EDR is found to be out of order.

> **Note:**
>
> If an out-of-order event comes in from a previous billing cycle after billing has run for that cycle (for example, when delayed billing is configured), the event is handled by default as if it is part of the current billing cycle.

How BRM Rerates Out-of-Order Events

The overall process for rerating out-of-order events is as follows:

1. Pipeline Manager loads your detection criteria configuration at startup. See "About Detecting Out-of-Order Events".
2. An event is rated in the pipeline and is rated by the rating and discounting modules.
3. The FCT_EventOrder module gets the out-of-order detection criteria from the DAT_PortalConfig module to determine if the event needs to be rerated.

   In addition, FCT_EventOrder gets data from the DAT_AccountBatch module that specifies the latest event processed time for each appropriate criterion and balance group combination for an active billing cycle.
4. FCT_EventOrder determines whether the event needs to be rerated:
   - If the event is out of order, the module proceeds to the next step.
   - If the event is not out of order, FCT_EventOrder triggers an update to the last event processed time in the DAT_AccountBatch memory. A record (record type = 850) is added to the pipeline output to update the **/profile/event_ordering** object with the new event's start time.
5. The event data record (EDR) is kept in the FCT_EventOrder module's shared memory until after the transaction commits, when it is then sent to a rerate-request file. When Batch Controller detects this file, it starts the OODHandler batch handler to process it. The rerate-request file contains multiple accounts for which out-of-order events were detected.

> **Note:**
>
> - You can configure FCT_EventOrder to write the out-of-order EDR data to separate rerate-request files after the transaction commits; batching the requests in this way helps performance in Pipeline Manager. See "Batching Out-of-Order Rerate Jobs".
> - You can configure FCT_EventOrder to write a specific number of accounts to each rerate-request file. See "Configuring the Number of Accounts in an Out-of-Order Rerate Job".

Rerate-request file names use the following format:

outputPrefix_pipelineName_transactionID_sequenceNumber.**xml**

where:

outputPrefix is the prefix specified by the **OutputPrefix** entry in the FCT_EventOrder module registry. The default is **ood**.

pipelineName is the name of the pipeline; in the following example, the name is ALL_RATE.

transactionID is the transaction ID.

sequenceNumber is the sequence number of the job.

For example:

```
ood_ALL_RATE_14_0.xml
```

6. The OODHandler batch handler processes the out-of-order rerate-request file, moves it to the *BRM_home***/apps/pin_ood_handler/process** directory, and calls the **pin_load_rerate_jobs** utility.
7. The **pin_load_rerate_jobs** utility creates an input flist with the following rerate output file data:
   - Account
   - Balance Group
   - CriteriaName
   - Rerate Start Time
   - Service

   It then calls the PCM_OP_ACT_USAGE opcode in calc-only mode, which generates a notification event (**/event/notification/activity/out_of_order**) that triggers the PCM_OP_ACT_HANDLE_OOD_EVENT opcode. PCM_OP_ACT_USAGE passes the data to PCM_OP_ACT_HANDLE_OOD_EVENT.
8. PCM_OP_ACT_HANDLE_OOD_EVENT calls the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode.
9. PCM_OP_RERATE_INSERT_RERATE_REQUEST checks for duplicate rerate jobs in the rerate-request file and calls other opcodes to create a rerate job out of the file.
10. The rated event is loaded into the BRM database by Rated Event (RE) Loader. If the event qualifies for out-of-order detection (and the event is in order), RE Loader updates the account's **/profile/event_ordering** object.

> **Note:**
>
> The **/profile/event_ordering** object stores data for the current and next billing cycles and for closed billing cycles. To clean up data for closed billing cycles, see "Purging Event Ordering Profile Data for Closed Billing Cycles".

About Out-of-Order Rerating Criteria

When the FCT_EventOrder module checks for out-of-order events, it uses the following criteria:

- The timestamp of the event. If the event is later than the latest event that is under consideration, the event is not out of order.
- The out-of-order detection criteria in which the event is defined in the **/config/event_order_criteria** object. If the event is not defined in any criteria in the **/config/event_order_criteria** object, it is assumed the event does not need to be rated in chronological order.
- The last event processed time for each balance group and the name of the trigger-dependant rerating criteria for that event for a billing cycle in the account's **/profile/event_ordering** object. If the event has a later timestamp than the latest event processed time for an event that uses the same balance group and criteria name, the event is not out of order.

Define the criteria for the events you must rate in chronological order in the **/config/event_order_criteria** object. The data stored in the **/config/event_order_criteria** object consists of two parts:

- The name of the criterion.
- One or more parameters, such as a service or event type or a charge offer name or discount name.

Your configuration can use as many parameters as you require. Table 64-2 provides examples of out-of-order criteria and what they mean for out-of-order detection:

Table 64-2 Criteria for Out-of-Order Detection

Criteria	Out-of-Order Detection
Criteria name: GSM_TEL_SERVICE Parameter: **/service/gsm/telephony**	All events for the GSM telephony service should be considered for out-of-order rerating.
Criteria name: GSM_EVENTS Parameter: **/event/delayed/session/telco/gsm/telephony** Parameter: **/event/delayed/session/telco/gsm/sms** Parameter: **/event/delayed/session/telco/gsm/fax**	All GSM events should be considered for out-of-order rerating.
Criteria name: GSM_Product Parameter: GSM Voice Product	All events for a specific charge offer should be considered for out-of-order rerating. All EDRs that are rated by the GSM Voice Charge Offer will be considered for out-of-order rerating.

For detailed information on defining your criteria, see "Defining Out-of-Order Criteria".

Setting Up Out-of-Order Rerating

To set up out-of-order rerating, perform the tasks in the following sections:

- Defining Out-of-Order Criteria
- Loading Out-of-Order Criteria

> **Note:**
>
> Accounts created *after* you load the **/config/event_order_criteria** object in the BRM database qualify for out-of-order detection.

- Configuring Out-of-Order Detection in a Pipeline.
- Configuring Event Notification for Out-of-Order Rerating
- (Optional) "Specifying a Reason Code for Rerating Out-of-Order Events"
- Configuring Batch Controller for Rerating Out-of-Order Events
- Configuring the OODHandler Batch Handler for Rerating Out-of-Order Events
- (Optional) "Purging Event Ordering Profile Data for Closed Billing Cycles"

Defining Out-of-Order Criteria

When you define out-of-order criteria, you specify parameters for events, services, discount names, or charge offer names that qualify for out-of-order detection. The parameters you specify are the criteria BRM uses to ensure that events are rated in chronological order when you want them to be. For more information, see "About Out-of-Order Rerating Criteria".

> **Note:**
>
> You should know how your offers are structured to track balance elements, such as balance elements for a specific service or balance elements for a specific type of usage. Tracking balance elements can involve using service-level balance groups and a criterion is set up for events that use the same balance group.

You typically list parameters that use the same balance group under the same criterion name. Services that use different balance groups are listed under separate criterion names. The **/profile/event_ordering** object stores the latest event processed time for each appropriate criteria and balance group combination for each active billing cycle. For more information, see "Defining Criteria under Separate Criteria Names".

You define out-of-order criteria in the **pin_config_ood_criteria.xml** file in *BRM_home***/sys/data/config**. For a sample of this file, see "Sample Out-of-Order Criteria File".

You can define criteria by using one or more parameters.

- The simplest criteria uses a single parameter.

  For example:

```
- <OodCriteriaElement>
<CriteriaName>GSM_TEL_SERVICE</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/gsm/telephony</Parameter>
</ParameterList>
</OodCriteriaElement>
```

- You can use two parameters in the following combinations:
  - Service Type, Event Type
  - Service Type, Product
  - Service Type, Discount
  - Event Type, Product
  - Event Type, Discount

  For example:

```
- <OodCriteriaElement>
<CriteriaName>GSM_TEL_SERVICE</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/gsm/telephony</Parameter>
<Parameter Type="Event">/event/delayed/session/gsm/telephony</Parameter>
</ParameterList>
</OodCriteriaElement>
```

- You can use multiple parameters to combine service type, event type, charge offer, and discount.

  For example:

```
- <OodCriteriaElement>
<CriteriaName>GSM</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/telephony</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/sms</Parameter>
<Parameter Type="Product">GSM Voice Product</Parameter>
<Parameter Type="Product">GSM Voice Discount</Parameter>
</ParameterList>
</OodCriteriaElement>
```

If you use a combination of event, service, and charge offer or discount, the EDR parameters must match all of the following:

- One service type (for example, service type 1 or service type 2)
- One event type (for example, event type 1 or event type 2)
- One charge offer or discount offer (for example, charge offer 1 or charge offer 2 or discount offer 1 or discount offer 2)

For example, in the following detection criterion named GSM_Multi:

```
- <OodCriteriaElement>
<CriteriaName>GSM_Multi</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/telephony</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/sms</Parameter>
<Parameter Type="Product">GSM Voice Product Telephony</Parameter>
<Parameter Type="Discount">GSM Voice Product SMS</Parameter>
```

```
</ParameterList>
</OodCriteriaElement>
```

The EDR must match the following parameters:

- Service type **/service/telco/gsm/telephony**
- Event type **/event/delayed/session/telco/gsm/telephony**
- Charge Offer GSM Voice Charge Offer Telephony

Or the following parameters:

- Service type **/service/telco/gsm/sms**
- Event type **/event/delayed/session/telco/gsm/sms**
- Charge Offer GSM Voice Charge Offer SMS

That means that when an EDR arrives for that particular service rated by that particular charge offer when that particular event occurs, BRM considers it for out-of-order rerating.

Defining Criteria under Separate Criteria Names

List services that use the same balance group under the same criteria name. BRM does not differentiate between the events of these services when obtaining the latest event processed time in the account's profile object because they consume balance elements in the same way for rating.

The following out-of-order criteria would apply to two GSM services that share the same balance group:

```
- <OodCriteriaElement>
<CriteriaName>GSM</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
</ParameterList>
</OodCriteriaElement>
```

If the latest event processed time is later than an EDR event timestamp, the event needs to be rerated regardless of the service to which it belongs; for example, for the preceding GSM criteria, if the latest event processed time is 2 p.m. for an SMS event and the EDR is a telephony event with a 1 p.m. timestamp, the event is out of order and needs to be rerated.

If an account can own more than one instance of a service and each instance uses a different balance group, out-of-order rerating is applied to the balance group of the service instance to which the event belongs. Accounts can have multiple balance groups if you offer packages in which service-level balance groups are used. When events for these services must be rated in chronological order, you must evaluate the out-of-order criteria for these services separately from each other.

You list services that use different balance groups under separate criteria names. The following out-of-order criteria would apply to two GSM services that use different balance groups:

```
- <OodCriteriaElement>
<CriteriaName>GSM_TEL</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
</ParameterList>
</OodCriteriaElement>
- <OodCriteriaElement>
<CriteriaName>GSM_SMS</CriteriaName>
```

```
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
</ParameterList>
</OodCriteriaElement>
```

Because the latest event processed times for all services used by an account are tracked in one profile object, BRM uses the unique criteria name to differentiate between the service-specific latest event processed times. Thus, in the preceding criteria, BRM uses the criteria name GSM_TEL in the account's profile object to track the latest event processed time for the telephony events and uses the criteria name GSM_SMS to track the latest event processed time for the SMS events.

There may be times when you must define parameters under separate criterion names when they share the same balance group but you are tracking different balance elements. For example, if a GSM telephony service and a GSM discount share a balance group and are rated by the same event, you would define a criterion for the discount if you are tracking the last time balance elements were consumed with that particular discount.

Avoiding Overlapping Criteria

If you configure criteria already contained in another criteria, the **load_pin_config_ood_criteria** utility reports an error, and nothing is loaded into the database. In the following example, the GSM_TEL_1 criteria is a superset of the GSM_TEL_2 criteria, so GSM_TEL_2 is considered an overlapping criteria:

```
- <OodCriteriaElement>
<CriteriaName>GSM_TEL_1</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
</ParameterList>
</OodCriteriaElement>
- <OodCriteriaElement>
<CriteriaName>GSM_TEL_2</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/telephony</Parameter>
</ParameterList>
</OodCriteriaElement>
```

In the following example, the GSM_Dual_A criteria is a superset of the GSM_Multi_B criteria, so GSM_Multi_B is an overlapping criteria:

```
- <OodCriteriaElement>
<CriteriaName>GSM_Dual_A</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
<Parameter Type="Product">GSM Voice Product Telephony</Parameter>
<Parameter Type="Discount">GSM Voice Discount SMS</Parameter>
</ParameterList>
</OodCriteriaElement>
- <OodCriteriaElement>
<CriteriaName>GSM_Multi_B</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/telephony</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/sms</Parameter>
<Parameter Type="Product">GSM Voice Product Telephony</Parameter>
<Parameter Type="Discount">GSM Voice Discount SMS</Parameter>
```

```
</ParameterList>
</OodCriteriaElement>
```

You would choose one of the preceding criterion based on your goals. You would never use both of them. For example:

- You would configure the GSM_Dual_A criterion if you wanted BRM to consider EDRs for out-of-order rerating when they matched the following:
  - Any event type for the GSM telephony service that is also rated by the GSM Voice Charge Offer Telephony charge offer.

  or

  - Any event type for the GSM SMS service that is also rated by the GSM Voice Discount SMS discount.
- You would configure the GSM_Multi_B criterion if you wanted BRM to consider EDRs for out-of-order rerating when they matched the following:
  - The specific event type **/event/delayed/session/telco/gsm/telephony** for the GSM telephony service that is also rated by the GSM Voice Charge Offer Telephony charge offer.

  or

  - The specific event type **/event/delayed/session/telco/gsm/sms** for the GSM SMS service that is also rated by the GSM Voice Discount SMS discount.

Because GSM_Dual_A is configured to consider all event types including the specific event types configured in GSM_Multi_B, the out-of-order detection configured in GSM_Multi_B is already handled by the GSM_Dual_A criterion. If you only want those specific event types to be checked for out-of-order detection, you would use GSM_Multi_B rather than GSM_Dual_A.

Sample Out-of-Order Criteria File

The following is a sample configuration of the **pin_config_ood_criteria.xml** file.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <OodCriteriaConfiguration xmlns="http://www.portal.com/schemas/BusinessConfig"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
www.portal.com/schemas/BusinessConfig pin_config_ood_criteria.xsd">

- <!--  This file is a sample file. -->
- <!--  This file needs to be modified based on the out-of-order configuration required
-->
- <!--  for an installation. THIS FILE SHOULD NOT BE LOADED WITH THE DATA SUPPLIED HERE.
-->

- <OodCriteriaElement>
<CriteriaName>TEL_SERVICE</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/telephony</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/telephony</Parameter>
</ParameterList>
</OodCriteriaElement>

- <OodCriteriaElement>
<CriteriaName>GSM_SERVICE</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm/sms</Parameter>
<Parameter Type="Event">/event/delayed/session/telco/gsm/sms</Parameter>
</ParameterList>
</OodCriteriaElement>
```

```
- <OodCriteriaElement>
<CriteriaName>A_MULTI_TEL_SERVICE</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm</Parameter>
<Parameter Type="Product">Standard GSM Telephony</Parameter>
</ParameterList>
</OodCriteriaElement>

- <OodCriteriaElement>
<CriteriaName>A_MULTI_TEL_SERVICE1</CriteriaName>
- <ParameterList>
<Parameter Type="Service">/service/telco/gsm</Parameter>
<Parameter Type="Discount">Standard GSM Discount</Parameter>
</ParameterList>
</OodCriteriaElement>

- <OodCriteriaElement>
<CriteriaName>TEL_SERVICE_EVENT</CriteriaName>
- <ParameterList>
<Parameter Type="Event">/event/delayed/session/telco/gsm</Parameter>
</ParameterList>
</OodCriteriaElement>

</OodCriteriaConfiguration>
```

Loading Out-of-Order Criteria

To configure out-of-order criteria, you edit the **pin_config_ood_criteria.xml** file and load the contents of the file into the BRM database by using the **load_pin_config_ood_criteria** utility. The data is stored in the **/config/event_order_criteria** object.

> **Note:**
>
> To connect to the BRM database, the **load_pin_config_ood_criteria** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

> **Note:**
>
> The **load_pin_config_ood_criteria** utility overwrites existing out-of-order criteria. If you are updating out-of-order criteria, you cannot load new out-of-order criteria only. You must load complete sets of out-of-order criteria each time you run the **load_pin_config_ood_criteria** utility.

> **Note:**
>
> You can run this utility to configure out-of-order criteria for different brands.

1. Edit the **pin_config_ood_criteria.xml** file in *BRM_home*/**sys/data/config**. For a sample of this file, see "Sample Out-of-Order Criteria File".

2. Save and close the file.
3. Use the following command to run the **load_pin_config_ood_criteria** utility:

   ```
   load_pin_config_ood_criteria pin_config_ood_criteria.xml
   ```

   If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

   ```
   load_pin_config_ood_criteria BRM_home/sys/data/config/pin_config_ood_criteria.xml
   ```

   > **Note:**
   >
   > If you copy the **pin_config_ood_criteria.xml** file to the directory from which you run the **load_pin_config_ood_criteria**utility, you do not have to specify the path or file name. By default, the file is named **pin_config_ood_criteria.xml**. You can change this name.

4. If Pipeline Manager is running, send a CBPReload semaphore to reload data. The **Reload** semaphore is used by the DAT_PortalConfig data module. If Pipeline Manager is not running, it will load the data the next time it is started.

To verify that the out-of-order criteria were loaded, you can display the **/config/event_order_criteria** object by using Object Browser, or use the **robj** command with the **testnap** utility.

Configuring Out-of-Order Detection in a Pipeline

To configure out-of-order detection in a pipeline, you do the following:

1. Configure the DAT_PortalConfig module.
2. Configure the FCT_EventOrder module.

   > **Note:**
   >
   > FCT_EventOrder should be located in the pipeline *after* the rating (FCT_MainRating) and discounting (FCT_Discount) modules and *before* the rejection (FCT_Reject) module.

   When configuring FCT_EventOrder, specify the following:

   - The amount of data to include in each rerate request file. See "Batching Out-of-Order Rerate Jobs".
   - The number of accounts assigned to each rerate job. See "Configuring the Number of Accounts in an Out-of-Order Rerate Job".

Batching Out-of-Order Rerate Jobs

You can configure the FCT_EventOrder module to batch the out-of-order EDR data in its transactional memory and write it to separate rerate-request files after the transaction commits. Batching rerate jobs in this way improves performance during rerating. To control how many records FCT_EventOrder batches into separate rerate request files, you specify an amount of time in minutes using the **RerateDelayTolerance** registry entry.

When the transaction commits, FCT_EventOrder sorts the rerate-request data in its transactional memory based on the EDR start time. The rerate start time of the first record in the rerate-request file is the rerate start time for the rerate job. As FCT_EventOrder writes the events into the rerate-request file, it compares the start time of the first record with the start time of each event being added. If the time difference exceeds the value of the **RerateDelayTolerance** entry, that event becomes the last record in the rerate-request file, and a new rerate-request file is created.

Example 1

If the **RerateDelayTolerance** entry is set to 180 minutes, and the start time of the first event in the rerate-request file is 3:15 p.m, events are added until the start time of an event is greater than 6:15 p.m.

Example 2

If the **RerateDelayTolerance** entry is set to 30 minutes, and the start time of the first event in the rerate-request file is 11:00 a.m. and four subsequent events have start times of 11:10 a.m., 12:05 p.m., 12:15 p.m., and 12:33 p.m, FCT_EventOrder creates two rerate-request files, resulting in two rerate jobs (**/job_batch/rerate** objects) as follows:

- Rerate job 1 contains accounts associated with EDRs that have these start times:

  11:00 a.m.

  11:10 a.m.

- Rerate job 2 contains accounts associated with EDRs that have these start times:

  12:05 p.m.

  12:15 p.m.

  12:33 p.m.

In the preceding example, if for one account the first EDR arrives at 11:10 a.m. and the last EDR arrives at 11:45 a.m., the account is included in Rerate job 1.

Configuring the Number of Accounts in an Out-of-Order Rerate Job

To improve batch rerating throughput, you can specify the number of accounts FCT_EventOrder assigns to each rerate job by using the **NumberOfAccountLimit** registry entry. FCT_EventOrder writes out the number of accounts specified by this entry to the rerate-request file, which, in turn, becomes the rerate job after detection of duplicate rerate requests is complete.

The default for the **NumberOfAccountLimit** registry entry is 1000.

> **Note:**
>
> The **NumberOfAccountLimit** registry entry of FCT_EventOrder is similar to the **per_job** configuration entry of the **pin_rerate** utility. When you configure these entries, BRM recommends you use the same value.

For more information, see "Configuring the Number of Accounts Per Job and Number of Jobs per Transaction".

Configuring Event Notification for Out-of-Order Rerating

To create rerate jobs automatically when certain events are not in chronological order, you must configure event notification for out-of-order rerating. BRM uses the **/event/notification/activity/out_of_order** event to trigger automatic rerating when an out-of-order event occurs.

To configure event notification for out-of-order rerating, do the following:

1. If your system has multiple configuration files for event notification, merge them.

   All of the event notification configuration files available in your system are in the *BRM_home***/sys/data/config** directory.

   Depending on which BRM features you use, your system may contain one or more configuration files for event notification; for example, a wireless system may use **pin_notify_ifw_sync** (supports Account Synchronization and standard recycling) or **pin_notify_telco** (supports GSM Manager). If your system contains more than one of these files, you must merge their contents into a single file.

2. Add the following entry to the merged file:

   ```
   # Event notification for out-of-order rerating
   177    0    /event/notification/activity/out_of_order
   ```

   This configures the **/event/notification/activity/out_of_order** event to trigger rerating and to call the PCM_OP_ACT_HANDLE_OOD_EVENT opcode (opcode ID number 177) to select which events to rerate.

   PCM_OP_ACT_HANDLE_OOD_EVENT calls the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode to insert a rerate job for the out-of-order events.

   > **Note:**
   >
   > You can configure this notification event to call a custom opcode. The custom opcode can analyze the event, determine if rerating is required, and then call PCM_OP_RERATE_INSERT_RERATE_REQUEST to create the rerate job with a rerate reason. For more information, see "Setting Up Trigger-Dependent Rerating".

3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list.
4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger.
5. Load your final event notification list into the BRM database by running the **load_pin_notify** utility to load the contents of the file into the **/config/notify** object.

Specifying a Reason Code for Rerating Out-of-Order Events

Rerate jobs can be assigned a rerate reason code. When you run the **pin_rerate** utility, you can rerate all accounts for a rerate job with a specific reason code. By default, rerate jobs for out-of-order events have the reason code **1**. You can change this to any number you want in the CM **pin.conf** file by using the following entry:

```
fm_act ood_rerate_job_reason_code rerate_reason_code
```

> **Note:**
>
> Rerate reason codes can also be assigned by other BRM automatic rerating features and when you run **pin_rerate**. To rerate accounts according to type of rerate reason code, ensure that your rerate reason codes are unique.

Configuring Batch Controller for Rerating Out-of-Order Events

The OODHandler batch handler is run automatically by Batch Controller. You must configure Batch Controller and the OODHandler batch handler.

To configure Batch Controller to rerate out-of-order events:

1. Open the Batch Controller **Infranet.properties** file in *BRM_home***/apps/batch_controller**.
2. Add the following entries listed in Table 64-3 for the **OODHandler** batch handler:

Table 64-3 Entries for OODHandler Batch Handler

Entry	Description
batch.random.events	Specify your event identifier in the event identifier list. For example: `batch.random.events TEL, SMS, FAX, OODHANDLER` where **OODHANDLER** is your event identifier.
*event_identifier***.name**	Specify a description for the event identifier. For example: `OODHANDLER.name OODHANDLER usage`
*event_identifier***.file.location**	Specify the path to the directory where Pipeline Manager puts the rerate request files. The location of this directory is configured in the **OutputDirectory** entry in the FCT_EventOrder module wireless registry file (*pipeline_home***/conf/wireless.reg**). **Important:** You must create the directory. It is not created by BRM installation scripts. For example: `OODHANDLER.file.location pipeline_home/data/out/ood`
*event_identifier***.file.pattern**	Specify the rerate job output file name. When Batch Controller detects a file with this name, it starts the batch handler. **Tip:** You can use an asterisk (*) to represent zero or more characters in the file name. No other wildcards are supported. For example: `OODHANDLER.file.pattern *.xml`

Table 64-3 (Cont.) Entries for OODHandler Batch Handler

Entry	Description
event_identifier.**handlers**	Specify the batch handler identifier. For example: `OODHANDLER.handlers OodHandler`
handler_identifier.**name**	Specify a description for the batch handler identifier. For example: `OodHandler.name OODHandler`
handler_identifier.**max.at.lowload.time** *handler_identifier*.**max.at.highload.time**	Specify the number of batch handler instances that can run concurrently during periods of low load and high load usage. Typical default settings are 6 at low load and 3 at high load. For example: `OodHandler.max.at.lowload.time 6` `OodHandler.max.at.highload.time 3`
handler_identifier.**start.string**	Specify the command that starts the OODHandler batch handler. For example, the default is: `OodHandler.start.string` *BRM_home*`/apps/pin_ood_handler/OODHandler.pl.` **Important:** Copy the *BRM_home*/**bin/OODHandler** to *BRM_home*/**apps/pin_ood_handler/OODHandler.pl**.

3. Save and close the file.

Configuring the OODHandler Batch Handler for Rerating Out-of-Order Events

The OODHandler batch handler retrieves the rerate-request file from the Pipeline Manager output and runs the **pin_load_rerate_jobs** utility to create rerate jobs. After the rerate jobs are created, the batch handler moves the rerate-request file to a different directory.

> **Note:**
>
> If you use the ConfigurableValidityHandler batch handler for loading first-usage validity data, do not use the **OODHandler_config.values** file as instructed below. Instead, you must configure the OODHandler batch handler in the ConfigurableValidityHandler configuration file (*BRM_home*/**apps/pin_rel/ConfigurableValidityHandler_config.values**). ConfigurableValidityHandler runs both the **pin_load_rerate_jobs** utility and the utility for loading validity data.

To configure the OODHandler batch handler:

1. Open the **OodHandler_config.values** configuration file in *BRM_home*/**apps/pin_ood_handler**.
2. Edit the following entries shown in Table 64-4. For information about other entries, see the **OodHandler_config.values** file.

Table 64-4 Entries to Configure OodHandler Batch Handler

Entry	Description
$FILETYPE	Specify the file name pattern of the rerate-request file. For example: `$FILETYPE = "*.xml.bc";` **Note:** The asterisk (*) represents zero or more characters in the file name. No other wildcards are supported. Batch Controller runs the OODHandler batch handler for each file with a name that matches this pattern. **Important:** The file name pattern must end with the **.bc** extension. Batch Controller automatically appends **.bc** to each file name before it runs a batch handler.
$HANDLER_DIR	Specify the path to the directory containing the OODHandler batch handler configuration files, log files, and other processing files and directories. The default is *BRM_home***/apps/pin_ood_handler**.
$pinLoadRerateJobsDir	Specify the path to the directory where the **pin_load_rerate_jobs** utility processes the files (this contains the **pin_rerate_job_info.xsd** file). The default is *BRM_home***/apps/pin_ood_handler/process**.
$pinLOADRERATEJOBS	Specify the application run by the OODHandler batch handler to process the files. For example: `$pinLOADRERATEJOBS = "pin_load_rerate_jobs";`
$STAGING	Specify the full path to the OODHandler batch handler rerate-request file location. For example: `$STAGING = "$pipeline_home/data/out/ood";` This is the same directory where Pipeline Manager puts the rerate-request files. When Batch Controller detects the rerate-request files in this location, it calls the OODHandler batch handler. **Note:** The location of this directory is also configured in the **OutputDirectory** entry in the FCT_EventOrder module wireless registry file (*pipeline_home***/conf/wireless.reg**). You must create the directory. It is not created by BRM installation scripts.
$PROCESSING	Specify the full path to the directory from which the rerate job files are processed by the OODHandler batch handler. The default is **$pinLoadRerateJobsDir**. The OODHandler batch handler takes the files from the **$STAGING** directory and places them here.
$LOGFILE	Specify the full path to the OODHandler batch handler log file. For example: `$LOGFILE = "$HANDLER_DIR/OOD_Handler.log";`

3. Save and close the file.

Purging Event Ordering Profile Data for Closed Billing Cycles

If out-of-order detection is configured in the system, one instance of the **/profile/event_ordering** object is created for each account during customer creation. The profile exists

for the lifetime of the account, and entries are updated and added to it by RE Loader every time events are processed for services belonging to the account that qualifies for out-of-order detection.

You can use the **purge_profile_event_ordering** script to clean up entries for closed billing cycles from the **/profile/event_ordering** object as follows:

Log on to the machine where **dm_oracle** is installed and run the Shell script:

```
BRM_home/apps/pin_rel/purge_profile_event_ordering
```

The **purge_profile_event_ordering** script is located in *BRM_home*/**apps/pin_rel**.

> **Note:**
>
> Cleaning up data for closed billing cycles is not automatically synchronized with Pipeline Manager. You must restart Pipeline Manager for the in-memory entries to be cleaned up.

About Trigger-Dependent Rerating

Trigger-dependent rerating enables you to specify when events are automatically rerated. For example, if you rerate usage following a charge offer cancellation, you can set up a charge offer cancellation event to automatically trigger rerating.

To configure trigger-dependent rerating, you set up event notification to call a custom opcode that analyzes events to determine if rerating is required. For example, the criteria for rerating might be:

- If the event is a cancel event.
- If the cancel event for a charge offer requires proration on cancellation.
- If rerating needs to occur to calculate proration for this charge offer.

If the notification event triggers rerating, the custom opcode specifies which events for an account must be rerated and sends the rerating requirements to the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode to create a rerate job.

Trigger-dependent rerating works as follows:

1. An event that you have configured in event notification occurs to call the custom opcode. For example, this might be a purchase event, charge offer cancellation, or change in account status. All the fields in the event are passed as input to the custom opcode.
2. The custom opcode analyzes the event using your custom selection criteria to determine if it should trigger rerating. For example, the opcode might be called when all purchase events occur, but not all purchase events will trigger rerating.
3. If the event triggers rerating because it matches your selection criteria, the custom opcode assigns an optional rerate reason code for rerating those events, specifies any price overrides, and calls PCM_OP_RERATE_INSERT_RERATE_REQUEST to create the rerate job.
4. PCM_OP_RERATE_INSERT_RERATE_REQUEST creates a rerate job that includes:
   - The account that needs to be rerated.
   - The events that must be rerated for that account.

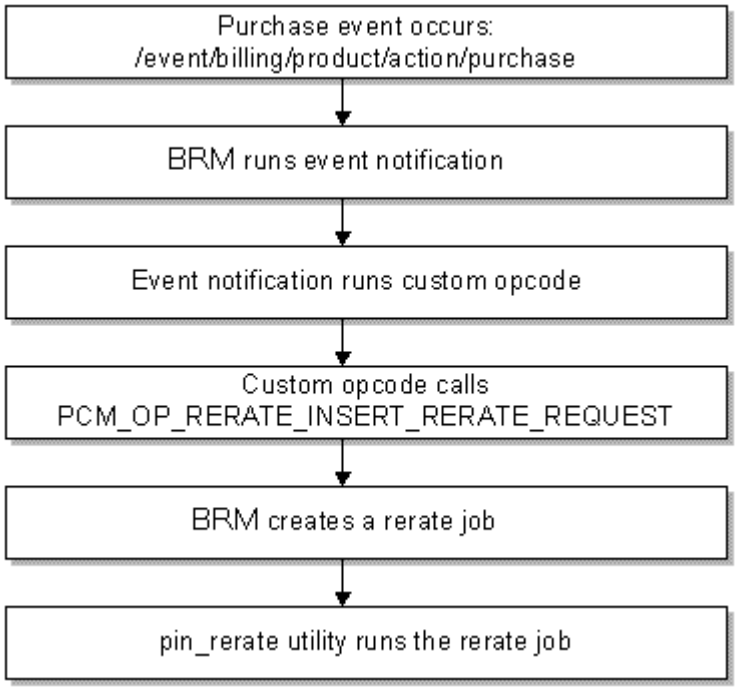
- The start time from which to rate the events.
- The rerate reason.
- Price overrides, if any.

> **Note:**
>
> If more than one account is included in the rerate job, the same selection criteria, price overrides, and start time apply to all the accounts.

5. You run the **pin_rerate** utility to process the rerate job and rerate the events.

Figure 64-2 shows the trigger-dependent rerating process when a purchase event occurs:

Figure 64-2 Trigger-Dependent Rerating Process



To set up trigger-dependent rerating, see "Setting Up Trigger-Dependent Rerating".

Setting Up Trigger-Dependent Rerating

To set up trigger-dependent rerating, you do the following:

1. Create a custom opcode for trigger-dependent rerating. See "Creating a Custom Opcode for Trigger-Dependent Rerating".
2. Configure event notification for trigger-dependent rerating. See "Configuring Event Notification for Trigger-Dependent Rerating".

Creating a Custom Opcode for Trigger-Dependent Rerating

To use trigger-dependent rerating, you must write custom code to specify when to create rerate jobs automatically when certain events occur. You can write your code by doing one of the following:

- Create one or more custom opcodes.

  You can create multiple custom opcodes to handle rerating events for different scenarios or create one opcode to handle all scenarios.

- Use the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST policy opcode.

  This policy opcode handles rerating events for BRM automatic rerating scenarios. You can modify and recompile this policy opcode to handle your own automatic rerating scenarios.

The opcode you use to define your custom code for trigger-dependent rerating must call the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode.

Your custom code is required to pass in the following to PCM_OP_RERATE_INSERT_RERATE_REQUEST:

- The POIDs of the accounts to be rerated.

  You can also pass in an optional array of additional accounts that are related to these accounts (for example, an account that used the account's service before a line transfer) and the start time for each account.

  > **Note:**
  >
  > If an array of accounts is passed in, the same selection criteria and price overrides apply to all the accounts.

- The time from which events must be rerated for the accounts.

Your custom code can optionally pass in the following to PCM_OP_RERATE_INSERT_RERATE_REQUEST:

- A rerate reason code to take advantage of rerating according to type of rerate job. See "Specifying a Rerate Reason Code".
- Selection criteria so that only those events that are required for rerating are identified for an account. See "Specifying Selection Criteria".
- Price overrides to substitute an account's subscribed pricing for alternate pricing during rerating. See "Specifying Price Overrides".

Your custom selection criteria and price override information for creating rerate jobs can include anything that can be passed as input to the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode. For detailed information about the array and field names that can be passed, refer to the PCM_OP_RERATE_INSERT_RERATE_REQUEST input and output flist specifications and the class specifications of the **/job/rerate** and **/job_batch/rerate** storable classes.

When an event triggers your custom opcode, your opcode may need to obtain data from other events in the database to obtain all of the information required for rerating.

Specifying a Rerate Reason Code

PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST or your custom opcode can pass an optional rerate reason code to PCM_OP_RERATE_INSERT_RERATE_REQUEST. The rerate reason code is stored in the rerate job and can be used to select jobs from the rerate queue for rerating using **pin_rerate**.

Your rerate reason codes can be any integer, except 1 (1 is a reserved default value for pipeline-generated rerating requests).

> **Note:**
>
> Because you can use multiple rerate reason codes to group rerate jobs for different rerating scenarios, ensure that your rerate reason codes are unique.

The default reason code is **0**.

Specifying Selection Criteria

When an account needs to be rerated, not every event for that account may need to be rerated from the specified rerate start time. Your custom opcode can optionally send selection criteria to the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode so that all events for the account can be filtered to select only the subset of events that needs to be rerated.

For example, you may need to rerate events only when they are generated by a specific service or only when they are rated by a specific charge offer or discount offer. You can specify selection criteria to rerate all accounts' events that have an event type of **/event/delayed/session/telco/gsm/telephony** but only when they are for the service type **/service/telco/gsm/telephony** and only when they are rated by the charge offer GSM Voice Telephony.

Your selection criteria can be any of the following:

- Event types (array PIN_FLD_EVENTS)
- Service types (array PIN_FLD_SERVICES)
- Charge offer POIDs (array PIN_FLD_PRODUCTS)
- Discount POIDs (array PIN_FLD_DISCOUNTS)
- Bundle POIDs (array PIN_FLD_DEALS)

> **Note:**
>
> Charge offers and discount offers are mutually exclusive with bundles. Bundles include charge offers and discount offers, so you can pass in a bundle array instead of the charge offer array and discount array.

Specifying Price Overrides

You can specify charge offers, discount offers, or bundles to use for a specific rerate job to override an account's subscribed charge offer, discount offer, or bundles during rerating. The

price override applies only to that rerate job; subsequent rerate jobs use the subscribed pricing or their own override pricing.

Your custom opcode can send the following price override data to the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode to set a price override for a rerate job:

- An optional array of override charge offer POIDs along with the charge offer it is overriding (PIN_FLD_OVERRIDE_PRODUCTS).
- An optional array of override discount POIDs along with the discount it is overriding (PIN_FLD_OVERRIDE_DISCOUNTS).
- An optional array of override bundle POIDs along with the bundle it is overriding (PIN_FLD_OVERRIDE_DEALS).

> **Note:**
>
> Charge offers and discount offers are mutually exclusive with bundles because bundles include charge offers and discount offers. The bundle is translated to a list of charge offers and discount offers when it is passed to the rerating opcodes.

The override charge offers and discount offers or bundles must already be defined in your BRM database. They must also be functionally equivalent to the subscribed charge offers and discount offers or bundles; for example, the priority or event map would be the same. In addition, override charge offers and discount offers cannot be configured as first usage charge offers.

> **Note:**
>
> You must rerate accounts when an override pricing charge offer is canceled in a given billing cycle so that refunds can be applied correctly. See "Configuring Event Notification for Override Pricing".

BRM creates a record when rerating is run with price overrides by using the **/rerate_session/override_info** object. When override pricing is passed to the PCM_OP_SUBSCRIPTION_RERATE_REBILL opcode, this opcodes creates a **/rerate_session/override_info** object to capture the override pricing information used during rerating.

Price overrides are automatic when you use the Best Pricing feature. For that feature, BRM calculates the best price from alternate charge offers and discount offers and rerates events to use the best pricing. When rerating events, the charge offer or discount offer that provides the best pricing is the override charge offer or discount offer.

Configuring Event Notification for Trigger-Dependent Rerating

When you configure event notification for trigger-dependent rerating, you specify the following in the **pin_notify** file in *BRM_home***/sys/data/config** (or your own merged event notification configuration file):

- The events that trigger rerating.

- The custom opcodes you want BRM to use to analyze the events, to identify whether rerating is required.

To configure event notification for trigger-dependent rerating, do the following:

1. If your system has multiple configuration files for event notification, merge them.
2. Ensure that the merged file includes the events you want to trigger rerating and the opcode number of the custom opcode you want to analyze those events.

   For example, these entries call the custom opcode with opcode number 10000:

   ```
   # Event notification for trigger-dependent rerating
   10000    0    /event/customer/status
   10000    0    /event/billing/product/action/purchase
   ```

3. (Optional) Add, modify, or delete entries in your final event notification list.
4. (Optional) Create custom code for event notification to trigger.
5. Load your final event notification list into the BRM database.

Configuring Event Notification for Override Pricing

If you use override pricing, you must set up trigger-dependent rerating to rerate accounts for cases where a refund could not be applied because an override pricing charge offer was canceled in a given billing cycle.

For a given billing cycle, if you rerate an account with an override charge offer and the charge offer is canceled, BRM cannot calculate a refund for that account because the charge offer is not available to apply the necessary cycle fees. When the refund cannot be applied, BRM creates the notification event **/event/notification/product/cancel/no_refund**.

To calculate the correct refund amount, you must set up trigger-dependent rerating as follows:

1. Write custom code to specify:
   - The account to rerate from the **/event/notification/product/cancel/no_refund** event.
   - The charge offer to use to apply the cycle fees for the refund to use the override charge offer that was canceled.

     BRM uses the base pricing charge offer associated with the account if you do not specify the override charge offer that was canceled. To specify the override charge offer that was canceled, obtain its information from the latest **/rerate_session/override_info** object created for the given account.
2. Include your code in a custom opcode or in the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST policy opcode that handles automatic rerating (opcode number 3787).

   If you create a custom opcode, it must call the PCM_OP_RERATE_INSERT_RERATE_REQUEST opcode to create the rerate job.
3. Set up event notification for the event **/event/notification/product/cancel/no_refund** to call your custom opcode or the policy opcode. For example, to call the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST policy opcode, you would enter:

   ```
   # Event notification for trigger-dependent rerating
   3787    0    /event/notification/product/cancel/no_refund
   ```

   For detailed instructions on setting up event notification, see "Configuring Event Notification for Trigger-Dependent Rerating".

65
Using the pin_rerate Utility

This chapter provides an overview of how you use the Oracle Communications Billing and Revenue Management (BRM) **pin_rerate** utility and the functions the utility performs.

About Using the pin_rerate Utility

You use the **pin_rerate** command-line utility to perform the following tasks:

- Select accounts and events for rerating from the BRM database. When accounts are selected, **pin_rerate** creates rerate jobs for the selected accounts. See "Selecting Accounts and Events for Rerating".

  You can assign a rerate reason code to the jobs created for the selected accounts and events. This enables you to rerate the accounts separately based on the reason for rerating. See "Assigning Rerate Reasons to Rerate Jobs".

  You can also define custom **pin_rerate** parameters based on various event criteria. This enables you to further customize which event attributes are used to select accounts and events for rerating. See "Defining Custom pin_rerate Parameters for Rerating".

- Rerate the accounts. To rerate accounts, you process rerate jobs. The process for rerating depends on whether you rerate real-time-rated and pipeline-batch-rated events concurrently or, if you do not use Pipeline Manager for batch rating, rerate only real-time-rated events. See the following sections:

  – Rerating Real-Time-Rated and Pipeline-Rated Events Concurrently

  – Processing Rerate Jobs When You Do Not Use Pipeline Batch Rating

  For information about processing jobs that are automatically created by BRM automatic rerating features, see "About Processing Rerate Jobs Created by Automatic Rerating".

  > **Note:**
  >
  > Do not move accounts to another database schema while rerating events for those accounts.

- Reprocess any rerate jobs that failed. See "Processing Failed Rerate Jobs".
- Back out the balance impacts of rating without rerating the events. See "Using pin_rerate for Back-Out-Only Rerating".
- Generate reports about the results of rerating. See "Reports Generated by the pin_rerate Utility".

If rerating fails, **pin_rerate** creates a report that includes the account numbers and start times for failed rerate jobs. The report file name is **pin_rerate.status_report** and is in the directory from which you ran the utility.

When rerate jobs have been processed, you can run **pin_rerate** to purge them from the database.

Selecting Accounts and Events for Rerating

The first step in the rerating process is running **pin_rerate** to select the accounts and associated events to rerate from the BRM database. The **pin_rerate** utility creates rerate jobs (**/job/rerate** and **/job_batch/rerate** objects) to store the information about the selected accounts.

> **Note:**
>
> If you use automatic rerating, accounts are selected and rerate jobs are created automatically for certain scenarios.

> **Note:**
>
> Do not move accounts to another database schema while rerating events for those accounts.

Specifying Accounts for Rerating

By default, the **pin_rerate** utility selects for rerating all the accounts and then all the events associated with those accounts that occurred from the start time that you specify.

For example, the following command selects all the accounts from the BRM database and, for those accounts, selects all the events that occurred after 07/23/2007.

```
pin_rerate -t 07/23/2007
```

pin_rerate provides a set of parameters that you can optionally use to select only specific accounts that meet one of the following requirements:

- One or a set of accounts identified by the account POIDs
- Accounts with events rated by a particular charge offer
- Accounts with events rated by a particular discount offer
- Accounts with events rated by charge offers and discount offers associated with a particular bundle
- Accounts with events generated for a particular service type or subscription service
- Accounts with events associated with an account's bill unit or bill unit and balance group
- Accounts that have particular event types

For example, when you use the **-p** parameter:

```
pin_rerate -p products.txt -t 07/23/2007
```

pin_rerate does the following:

- Selects only the accounts that have events related to the charge offers in the **products.txt** file.
- Selects all the events for the selected accounts that occurred after 07/23/2007.

When you use the **-line** parameter:

```
pin_rerate -line 6495832245 -t 09/21/04
```

pin_rerate does the following:

- Selects only the accounts that have the subscription service with the phone number 6495832245.
- Selects all the events for the selected accounts that occurred after 09/21/04.

Specifying Events for Rerating

By default, **pin_rerate** rerates all the events for the selected accounts from the start time that you specify. You can specify to rerate only specific events for the selected accounts by using the **-r** parameter. This is called *selective rerating*.

> **Note:**
>
> Do not use selective rerating if:
>
> - Your rating scheme includes credit limits or balance element-based tiers. These schemes require that all events related to an account are rerated to assure accurate rerating.
> - Deferred taxation is used for taxing events during rerating.

If you use selective rerating, be sure to consider how it might affect rating overall because account balances can be impacted by different types of events: a cycle event can grant free minutes and a usage event consumes free minutes from the same balance. If, for example, you change the pricing for a charge offer that grants free minutes, you must rerate all events for the accounts that own the charge offer. It would be incorrect to use selective rerating in this case.

The **-r** parameter must be used with parameters that specify the accounts to select for rerating. When the **-r** parameter is used, rerating applies the account selection criteria to the account's events as well and selects only those events that meet the selection criteria.

The **-r** parameter can be used with any of the account selection parameters.

For example, when you use **-r** with the **-s** parameter:

```
pin_rerate -r -s service.txt -t 07/23/2007
```

pin_rerate does the following:

- Selects only the accounts that have events related to the services in the **service.txt** file.
- Selects only the events related to the services in **service.txt** that occurred after 07/23/2007 for the selected accounts.

When you have a high volume of events to rerate, you can rerate events that are rated only in real time, such a cycle and purchase events. To do this, you use the **-r** parameter in combination with the **-n** parameter for specifying event types. You define all the event types rated by real-time rating in an input file.

For example, you could specify the following event types in a file named **event.txt**:

- **/event/billing/product/fee/cycle/cycle_forward_monthly**

- **/event/billing/product/fee/purchase**
- **/event/billing/product/fee/cycle/cycle_forward_arrear**

When you run the following command:

```
pin_rerate -t 01/01/2007 -n event.txt -r
```

pin_rerate does the following:

- Selects all accounts that have events with event types in the **event.txt** file.
- Selects only the events with event types in the **event.txt** file and which occurred after 01/01/2007 for the selected accounts.

When rerating cycle fee events, to get the correct rerating results, include the cycle events that occur during billing that are configured in the charge offer, such as cycle discount, rollover, and fold events in the event file.

For example, if a cycle discount is configured to be some percentage of the charge during billing and if the cycle forward arrear fee is modified during the billing cycle, then to rerate the cycle forward arrear event, you need to include both events in the event file:

- **/event/billing/product/fee/cycle/cycle_forward_arrear**
- **/event/billing/cycle/discount**

If the event type specified in the **-n** parameter input file has subclasses, all subclass events are also rerated, providing they meet the selection criteria. For example, if you specify **/event/delayed/session/telco** in the **-n** parameter input file, events of type **/event/delayed/session/telco/gsm** that meet the selection criteria are also rerated.

Customizing Event Searches for Selective Rerating

You can further refine the event selection criteria used for selective rerating by customizing the PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE policy opcode. This opcode is called when the **pin_rerate** utility is run with **-r** parameter to indicate selective rerating.

> **Note:**
>
> An alternative to customizing this policy opcode to filter the events selected for rerating is to create custom **pin_rerate** parameters instead. See "Defining Custom pin_rerate Parameters for Rerating".

PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE receives an event search template that is based on the account and event selection criteria specified in the **pin_rerate** command line: for example, to select events related to services specified by the **-s** parameter.

By default, PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE does not change the search template and returns a copy of the input flist in the output flist.

You can customize the search template in the input flist to rerate specific types of events. Most customizations include changes only to the fields listed in Table 65-1:

Table 65-1 Fields to Customize

Field	Description
PIN_FLD_TEMPLATE	The modified search template.
PIN_FLD_ARGS	The list of search arguments. **Note:** • This list must match the list of arguments in PIN_FLD_TEMPLATE. • It is preferable to have arguments of one type only. For example, an event search based on charge offer objects.

The PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE policy opcode receives the following fields in the RESULTS array from the PCM_OP_SUBSCRIPTION_RERATE_REBILL standard opcode:

- PIN_FLD_POID
- PIN_FLD_CREATED_T
- PIN_FLD_EFFECTIVE_T
- PIN_FLD_END_T
- PIN_FLD_SERVICE_OBJ
- PIN_FLD_ACCOUNT_OBJ
- PIN_FLD_UNRATED_QUANTITY
- PIN_FLD_RERATE_OBJ
- PIN_FLD_BAL_IMPACTS [*]
- PIN_FLD_SUB_BAL_IMPACTS [*]

> **Note:**
>
> – To assure that the existing mandatory fields in the array are passed back, avoid customizing the RESULTS array. Extra fields added to the array are retrieved but ignored by the standard opcode.
>
> – Test your customized template to ensure that it works properly before using it with a working database. Use the **-e** and **-c** parameters with the **pin_rerate** utility to test your modifications.

The PCM_OP_SUBSCRIPTION_POL_SPEC_RERATE policy opcode returns the search template that the PCM_OP_SUBSCRIPTION_RERATE_REBILL opcode uses to find events in selected accounts that need rerating.

Specifying the Event Sequence for Rerating

Events are rerated in sequence based on the event time. You can specify one of two times:

- *Event end time* defines the time that the usage actually occurred. This is the default.
- *Event creation time* is the time that the event was loaded into the BRM database.

The event time you specify might depend on how the original events were rated:

- **Events rated or loaded in batches**

  Events that are rated or loaded into the BRM database in batches have a significant delay between the event end time and creation time. Using the event end time reflects the actual real-time sequence of the original events. However, because batch events are recorded in order of creation time, this makes predicting the actual impact of a price configuration change harder. To compare the original and rerated balance impacts of batch events, use the event creation time.

- **Events rated and loaded in real time**

  Events that are rated and loaded in real-time have very little delay between the event end time and creation time. Both the event end time and creation time reflect the real-time sequence in which the original events occurred and were recorded.

By default, the **pin_rerate** utility rerates batch events in sequence based on event end time for both real-time and batch events.

To specify the event time for rerating, use the **-b** parameter when selecting accounts for rerating. The **-b** parameter takes either the **c** option (for event creation time) or the **e** option (for event end time).

For example:

```
pin_rerate -b c -t 07/23/2007
```

The preceding example selects all accounts, then selects both real-time and batch events that occurred after 07/23/2007 for the selected accounts, and finally rerates the events in order of creation time.

Assigning Rerate Reasons to Rerate Jobs

Some rerating jobs must be processed before others. For example, if a rerate job includes events that have an impact on future rating, such as volume-based discounting, those events must be rerated first. You can achieve this by assigning rerate reason codes to rerate jobs when you create those jobs. You can then select only those jobs associated with the specified rerate reason code during scheduled rerating executions.

Rerate reason codes can be any integer except **1** (**1** is reserved for pipeline-generated events that were rated out of order).

> **Note:**
>
> Ensure that your reason codes are unique to process rerate jobs associated with them during separate rerating executions.

You assign rerate reason codes when you manually create rerate jobs by using the **pin_rerate** utility.

> **Note:**
>
> Rerate jobs created automatically by BRM automatic rerating features are not assigned a reason code by default. You can assign rerate reason codes for trigger-dependent and automatic rerating by customizing the PCM_OP_SUBSCRIPTION_POL_GENERATE_RERATE_REQUEST policy opcode.
>
> You can assign a unique reason code for out-of-order rerating by using a configuration file.

To assign a rerate reason code by using **pin_rerate**, you specify the rerate reason code on the command line along with other account selection parameters, such as the start time and other parameters:

```
pin_rerate -reason reason_code -t start_time other_parameters
```

You can specify only one reason code when creating rerate jobs. Rerate jobs are created for the selected accounts and all rerate jobs are assigned the specified rerate reason code.

For example, to assign a rerate reason code of **99** to rerate jobs that rerate all account's events that occurred after January 1, 2007 and that are associated with the charge offers specified in the **product.txt** file, you enter the following command:

```
pin_rerate -reason 99 -t 01/01/2007 -p product.txt
```

Defining Custom pin_rerate Parameters for Rerating

You can define custom **pin_rerate** parameters based on any event criteria. This enables you to customize which event attributes are used to select accounts and events for rerating.

To define custom parameters, you map extraction keys to fields in the event object. You then run the **load_pin_rerate_flds** utility to load the extraction-key-to-event-field mappings into the **/config/rerate_flds** object in the BRM database. See "Configuring Custom pin_rerate Parameters".

When you run **pin_rerate**, it uses the extraction key to find the corresponding event field name in the **/config/rerate_flds** object. It uses the event field name to find and retrieve accounts and events for rerating.

For example, if you map the extraction key **resource_id** to the PIN_FLD_RESOURCE_ID field in the **/event** object, you can run **pin_rerate** with the following command, specifying the input file containing the balance element IDs:

```
pin_rerate -resource_id input_file -t start_time
```

In this example, **pin_rerate** selects accounts to be rerated that have events associated with the balance elements specified in the input file.

By default, BRM searches only the base **/event** class for fields associated with custom **pin_rerate** parameters. If the event field associated with a custom parameter is present only in a subclass, you must specify the subclass event type in the command line by including the **-n** parameter.

> **Note:**
>
> When used with custom **pin_rerate** parameters, the input file for the **-n** parameter can contain only one event type. If you specify more than one event type, an error is returned.

Configuring Custom pin_rerate Parameters

To configure custom **pin_rerate** parameters, you perform the following tasks:

- Defining Custom Parameters
- Loading Custom Parameters

Defining Custom Parameters

To define custom **pin_rerate** parameters, create an XML file that maps the parameters to event fields. You can map to fields specified in the EVENT_T and EVENT_BAL_IMPACTS_T tables in the base **/event** class or to fields in an event subclass.

> **Note:**
>
> - If you map parameters to fields in an **/event** subclass, you must specify the subclass by using the **-n** parameter when you run **pin_rerate** with the custom parameter.
> - To create the XML file, you must be familiar with XML and the XML schema.

Create an XML file containing the parameter-to-event field mappings according to the BRM rerating XML schema file (*BRM_home***/xsd/pin_rerate_flds.xsd**).

> **Note:**
>
> Validate your XML file against the XML schema. The **load_pin_rerate_flds** utility cannot successfully load an invalid XML file.

Sample XML parameter file

The following example shows how you specify event fields in **<Name>** tags and the parameters they map to in **<value>** tags:

```
<PinRerateList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.portal.com/InfranetXMLSchema pin_rerate_flds.xsd">
     <PinRerate>
          <Name>EVENT.PIN_FLD_START_T </Name>
          <value>startTime</value>
     </PinRerate>
     <PinRerate>
          <Name>EVENT.PIN_FLD_END_T</Name>
          <value>endTime</value>
```

```
    </PinRerate>
    <PinRerate>
         <Name>EVENT.PIN_FLD_ACCOUNT_OBJ</Name>
         <value>account</value>
    </PinRerate>
    <PinRerate>
         <Name>EVENT.PIN_FLD_RATE_PLAN </Name>
         <value>ratePlan</value>
    </PinRerate>
    <PinRerate>
         <Name>EVENT.PIN_FLD_DEAL </Name>
         <value>deal</value>
    </PinRerate>
    <PinRerate>
         <Name>EVENT.PIN_FLD_SERVICE_OBJ </Name>
         <value>service</value>
    </PinRerate>
    <PinRerate>
         <Name>EVENT.PIN_FLD_BAL_IMPACTS.PIN_FLD_PRODUCT_OBJ</Name>
         <value>product</value>
    </PinRerate>
</PinRerateList>
```

If you map a custom parameter to a field in an **/event** subclass, specify the array or substruct (if any) that contains the parameter field.

For example, to specify PIN_FLD_ORIGIN_NETWORK, which is located in the PIN_FLD_TELCO_INFO substruct in the **/event/delayed/session/telco** subclass:

```
<PinRerate>
<Name>EVENT.PIN_FLD_TELCO_INFO.PIN_FLD_ORIGIN_NETWORK</Name>
<value>origin_network</value>.
</PinRerate>
```

Loading Custom Parameters

Use the following command to load the custom **pin_rerate** parameters defined in the XML file into the **/config/rerate_fld** object:

BRM_home/**sys/data/config/load_pin_rerate_flds** *xml_file_name*

> **Note:**
>
> The **load_pin_rerate_flds** utility uses a configuration file (**pin.conf**) located in the same directory to connect to the BRM database. Edit the configuration file to connect to your BRM database.

About Processing Rerate Jobs Created by Automatic Rerating

Rerate jobs are created automatically by the following features:

- Automatic rerating.
- Trigger-dependent rerating.
- Out-of-order rerating.

These rerate jobs are automatically processed when you run the **pin_rerate** with one of the following parameters:

- The **-process jobs** parameter when rerating real-time-rated and pipeline-rated events concurrently.
- The **-rerate** parameter when rerating only real-time-rated if you do not use Pipeline Manager for batch rating.

By default, rerate jobs that are automatically created are not assigned a rerate reason code. If you customize automatic rerating to assign a specific rerate reason code to rerate jobs, you process those jobs by using the **-reason** parameter with the **-process jobs** or **-rerate** parameter.

For more information, see the following sections:

- Rerating Real-Time-Rated and Pipeline-Rated Events Concurrently
- Processing Rerate Jobs When You Do Not Use Pipeline Batch Rating

Rerating Real-Time-Rated and Pipeline-Rated Events Concurrently

When rerating real-time-rated and pipeline-rated events concurrently, you process rerate jobs in the following ways:

- By processing all existing rerate jobs that were previously created. See "Processing Rerate Jobs for Concurrent Rerating ".
- By processing only rerate jobs associated with a rerate reason code. See "Processing Rerate Jobs According to a Rerate Reason Code for Concurrent Rerating".

> **Note:**
>
> Ensure that no event data records (EDRs) are in the pipeline before you run **pin_rerate**. Otherwise, EDRs might be rated using old account data.

Processing Rerate Jobs for Concurrent Rerating

When rerating real-time-rated and pipeline-batch-rated events concurrently, rerate jobs are created and processed in separate commands. Rerate jobs can be created in the following ways:

- Automatically by BRM automatic rerating features.
- When you run **pin_rerate** to select accounts for rerating, as described in "Selecting Accounts and Events for Rerating". For example:

  ```
  pin_rerate -t start_time -s input_file
  ```

To process all existing rerate jobs in the rerate queue, run the following commands:

1. **pin_rerate -process jobs**. See "Notifying the Batch Pipeline That Rerating Is Starting".
2. **pin_rerate -process queue**. See "Processing Acknowledgment Events from the Batch Pipeline".

3. **pin_rerate -rerate**. See "Rerating the Events Associated with the Accounts in Rerate Jobs".
4. **pin_rerate -process queue**. See "Processing Acknowledgment Events from the Batch Pipeline".
5. **pin_rerate -process recycle**. See "Recycling EDRs Suspended during Rerating".

For example:

```
pin_rerate -process jobs
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

> **Note:**
>
> You run **pin_rerate** during non-peak hours with these commands as part of a cron job. See your operating system documentation for details on creating a cron job.

Notifying the Batch Pipeline That Rerating Is Starting

After rerate jobs are created, you run **pin_rerate** with the **-process jobs** parameter to notify the batch pipeline that rerating is about to begin.

When you specify the **-process jobs** parameter, the following actions are performed:

1. The **pin_rerate** utility finds all the rerate jobs with a status of NEW and sends business events containing the rerate job ID and the list of accounts associated with the rerate job to the batch pipeline.
2. The **pin_rerate** utility then changes the status of the jobs from NEW to WAITING_ACCOUNT_LOCKED to indicate that it is waiting for an acknowledgment from the pipeline.
3. The batch pipeline dequeues the business events and suspends rating EDRs for those accounts.
4. The batch pipeline sends an acknowledgment event back to the Account Synchronization database queue.

> **Note:**
>
> - If you process rerate jobs according to a rerate reason, the same actions are performed but only for the rerate jobs associated with the rerate reasons you specify.
> - To process rerate jobs according to a rerate reason by using the **pin_rerate** utility, see "Processing Rerate Jobs According to a Rerate Reason Code for Concurrent Rerating".

Processing Acknowledgment Events from the Batch Pipeline

You run **pin_rerate** with the **-process queue** parameter to process acknowledgment events sent by the batch pipeline.

You run **pin_rerate** with the **-process queue** parameter at two times:

- After you run **pin_rerate** with the **-process jobs** parameter. The acknowledgment event at this time indicates that the batch pipeline has suspended rating EDRs for the accounts in the rerate jobs. The **pin_rerate** utility dequeues the event and changes the status of the rerate jobs from WAITING_ACCOUNT_LOCKED to ACCOUNT_LOCKED.
- After you run **pin_rerate** with the **-rerate** parameter. The acknowledgment event at this time indicates that the batch pipeline has resumed rating EDRs for the accounts in the rerate job. **pin_rerate** dequeues the event and changes the status of the rerate jobs from RERATED to READY_FOR_RECYCLE.

Rerating the Events Associated with the Accounts in Rerate Jobs

When the rerate jobs has the ACCOUNT_LOCKED status, you run **pin_rerate** with the **-rerate** parameter to rerate the accounts.

pin_rerate finds all the rerate jobs with a status of ACCOUNT_LOCKED and calls rerating opcodes to rerate the accounts.

After an account is successfully rerated, **pin_rerate** sends a business event through the Account Synchronization Data Manager (DM) to the batch pipeline to update the account data. The DAT_BalanceBatch module processes the event and reloads the updated account balances from the BRM database.

When *all* the accounts in a rerate job have been successfully rerated and updated, **pin_rerate** updates the rerate job status from ACCOUNT_LOCKED to RERATED and then notifies the batch pipeline that rerating is complete. The batch pipeline resets the rerate flags for all the accounts in the rerate job and returns an acknowledgment event to inform **pin_rerate** that it has resumed rating EDRs for those accounts.

Recycling EDRs Suspended during Rerating

EDRs that are temporarily suspended by the batch pipeline are loaded into the BRM database by Suspended Event (SE) Loader. The suspended EDRs are stored in the database until they are recycled.

> **Note:**
>
> Before you can recycle suspended EDRs, they must be loaded into the BRM database by SE Loader. You typically schedule SE Loader to run automatically when you set up standard recycling.

To recycle the EDRs that were suspended during the rerating process, you run **pin_rerate** with the **-process recycle** parameter.

> **Note:**
>
> Suspended EDRs are typically recycled by running the **pin_recycle** utility. However, **pin_rerate** calls the standard recycling opcode directly so you do not use **pin_recycle** when using **pin_rerate**.

pin_rerate finds all the rerate jobs with a status of READY_FOR_RECYCLE and calls the standard recycling opcodes to recycle the associated EDRs. Standard recycling uses the recycle key value in the EDR to identify and retrieve the EDRs that were suspended during the rerating process.

After the EDRs are recycled, **pin_rerate** changes the status of the jobs from READY_FOR_RECYCLE to COMPLETE.

> **Note:**
>
> - In a multischema environment, you must run **pin_rerate** separately on each database schema.
> - If no EDRs were suspended for the accounts being rerated, the job status is still changed to COMPLETE.
> - If an error occurs while recycling EDRs, the job status is not changed; it retains the status of READY_FOR_RECYCLE.

Processing Rerate Jobs According to a Rerate Reason Code for Concurrent Rerating

To process rerate jobs according to a rerate reason code when rerating real-time-rated and pipeline-batch-rated events concurrently, you use the **-process jobs** parameter with the **-reason** parameter:

```
pin_rerate -process jobs -reason [reason_code_1,reason_code_2,reason_code_3,...]
```

where *reason_code_x* is the rerate reason code assigned to existing rerate jobs in one of the following ways:

- When you run **pin_rerate** with the **-reason** parameter to create rerate jobs and with an assigned rerate reason code. See "Assigning Rerate Reasons to Rerate Jobs".
- When you customized one of the following features to assign a specific rerate reason code to rerate jobs that are automatically created:
  - Automatic rerating.
  - Trigger-dependent rerating.
  - Out-of-order rerating.

You can list multiple reason codes separated by commas (do not use spaces between reason codes). For example, to process jobs with reason codes 100, 200, and 300, enter the following commands:

```
pin_rerate -process jobs –reason 100,200,300
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

> **Note:**
>
> When you list multiple rerate reasons, the rerate jobs associated with them are processed randomly. They are not processed in the order you list them in the command line, nor are they processed in increasing or decreasing numeric order. To process rerate jobs according to a rerate reason in a particular order, you must schedule separate rerating executions for them.

Processing Rerate Jobs When You Do Not Use Pipeline Batch Rating

If you do not use Pipeline Manager for batch rating, you process rerate jobs in the following ways:

- By creating rerate jobs and processing those jobs at the same time. See "Processing Rerate Jobs When Selecting Accounts for Rerating".
- By processing all existing rerate jobs that were previously created. See "Processing Existing Rerate Jobs ".
- By processing only rerate jobs associated with a rerate reason code. See "Processing Rerate Jobs per a Rerate Reason When Rerating Only Real-Time-Rated Events".

Processing Rerate Jobs When Selecting Accounts for Rerating

When rerating real-time events only, rerate jobs are processed when you run **pin_rerate** and specify the start time and any additional search criteria as described in "Selecting Accounts and Events for Rerating".

For example:

```
pin_rerate -t start_time -p input_file
```

The **pin_rerate** utility selects the accounts for rerating by using the search criteria, creates rerate jobs, and then immediately rerates the selected accounts.

Processing Existing Rerate Jobs

Some rerate jobs are automatically created by BRM automatic rerating features. Other rerate jobs can be precreated by assigning a rerate reason code when selecting the accounts for rerating.

To process all existing rerate jobs in the rerate queue, you use only the **-rerate** parameter:

```
pin_rerate -rerate
```

Processing Rerate Jobs per a Rerate Reason When Rerating Only Real-Time-Rated Events

To process rerate jobs according to a rerate reason when rerating only real-time-rated events, you use the **-rerate** parameter with the **-reason** parameter:

```
pin_rerate -rerate -reason reason_code_1,reason_code_2,reason_code_3,...
```

where *reason_code_x* is the rerate reason code assigned to existing rerate jobs in one of the following ways:

- When you run **pin_rerate** with the **-reason** parameter to create rerate jobs with an assigned rerate reason code. See "Assigning Rerate Reasons to Rerate Jobs".
- When you customized one of the following features to assign a specific rerate reason code to rerate jobs that are automatically created:
  - Automatic rerating.
  - Trigger-dependent rerating.
  - Out-of-order rerating.

You can list multiple reason codes separated by commas (do not use spaces between reason codes). For example, to process jobs with reason codes 100, 200, and 300, you enter the following commands:

```
pin_rerate -rerate –reason 100,200,300
```

> **Note:**
>
> When you list multiple rerate reasons, the rerate jobs associated with them are processed randomly. They are not processed in the order you list them in the command line, nor are they processed in increasing or decreasing numeric order. To process rerate jobs according to a rerate reason in a particular order, you must schedule separate rerating executions for them.

Processing Failed Rerate Jobs

To process failed rerate jobs, you run **pin_rerate** only with the commands for processing rerate jobs, without specifying selection criteria or rerate reason codes.

> **Note:**
>
> This also processes all rerate jobs that are in the rerate queue: not only failed rerate jobs.

For example:

- When rerating real-time-rated and pipeline-rated events concurrently, use the following commands:

```
pin_rerate -process jobs
pin_rerate -process queue
pin_rerate -rerate
pin_rerate -process queue
pin_rerate -process recycle
```

See "Rerating Real-Time-Rated and Pipeline-Rated Events Concurrently".

- If you do not use Pipeline Manager for batch rating, use the following command:

```
pin_rerate -rerate
```

See "Processing Rerate Jobs When You Do Not Use Pipeline Batch Rating".

Using pin_rerate for Back-Out-Only Rerating

Back-out-only rerating backs out the balance impacts of rating without rerating events.

BRM backs out balance impacts for back-out-only rerating by creating an adjustment event that fully negates the original balance impacts.

To back out the balance impacts of rating, run **pin_rerate** with the **-backout** parameter. Use the **-backout** parameter with other parameters that select the accounts and their events for rerating. This creates rerate jobs that are set for back-out-only rerating for the selected accounts.

For example, the following command selects accounts whose events were rated by the charge offers specified in the **products.txt** file and backs out those events rated by the charge offers that occurred after 06/01/2007:

```
pin_rerate -backout -r -p products.txt -t 06/01/2007
```

The **-r** parameter is used to select only the events that used the specified charge offers. Without the **-r** parameter, **pin_rerate** would select accounts whose events used the specified charge offers and then back out all events for those accounts.

> **Note:**
>
> Use caution when choosing the events to back out because it can impact your general ledger. For example, it is incorrect to use back-out-only rerating for a cycle event when the customer has already paid the cycle fee or to use back-out-only rerating when charge offer pricing is changed. Typically, back-out-only rerating is performed only on usage events where rating should not have occurred.

Using Custom pin_rerate Parameters with Back-Out-Only Rerating

You can perform back-out-only rerating using custom **pin_rerate** parameters. This enables you to select events for back-out-only rerating based on any event criteria.

For information about defining custom **pin_rerate** parameters, see "Defining Custom pin_rerate Parameters for Rerating".

By default, rerating searches only the base **/event** storable class for fields associated with custom **pin_rerate** parameters. If the event field associated with a custom parameter is present only in a subclass, you must also specify the subclass event type in the command line by using the **-n** *input_file* parameter.

If the event type specified in the **-n** parameter input file has subclasses, all subclass events are also backed out, providing they meet the selection criteria. For example, if you specify **/event/delayed/session/telco** in the **-n** parameter input file, events of type **/event/delayed/session/telco/gsm** that meet the selection criteria are also backed out.

Example of Using a Custom Parameter with Back-Out-Only Rerating

Suppose you define the custom parameter **origin_network** to select events based on the PIN_FLD_ORIGIN_NETWORK field. You specify an origin network ID in a file named **origin_network.txt**.

The PIN_FLD_ORIGIN_NETWORK field is located in the **/event/delayed/session/telco** subclass so you specify this subclass, in a file named **event_subclass.txt**.

To select accounts whose events were generated by the specified origin network and back out the balance impacts of only those events when they occurred after 06/01/2007, run the following command:

```
pin_rerate -backout -r -n event_subclass.txt -origin_network origin_network.txt -t
06/01/2007
```

Reports Generated by the pin_rerate Utility

By default, the rerating process generates summary and detailed reports. To print a report, use the **pin_rerate -l** parameter. You have the option of printing either a summary of the report, a detailed report, or both summary and detailed reports. If you do not specify which report to print, both summary and detailed reports are printed by default.

You set the reporting parameter as follows:

- When rerating both real-time-rated and pipeline-batch-rated events concurrently, you specify the **-l** parameter with the **-rerate** parameter. In the following example, the report option is set to print a summary report:

  ```
  pin_rerate -t 01/01/2007 -s service.txt
  pin_rerate -process jobs
  pin_rerate -process queue
  pin_rerate -rerate -l s
  pin_rerate -process queue
  pin_rerate -process recycle
  ```

  > **Note:**
  >
  > When the batch pipeline is enabled, you cannot specify the report option at rerate job creation time.

- If you do not use Pipeline Manager for batch rating, you specify the **-l** parameter at one of the following times:
  - When specifying the accounts to rerate. Enter the **-l** parameter after the parameters used to select the accounts. In the following example, the report option is set to print a detailed report:

    ```
    pin_rerate -t 01/01/2007 -p product.txt -l d
    ```

  - When processing rerate jobs that already exist, such as those created with a rerate reason code and those created by automatic rerating. Specify the **-l** parameter with the

-rerate parameter. In the following example, the report option is set to print a summary report:

```
pin_rerate -rerate -l s
```

> **Note:**
>
> Report generation is resource intensive and can degrade system performance, depending on system load and the number of events rerated. You can override report generation by using the **-l n** option with **-rerate** to skip printing reports. For example: **pin_rerate -rerate -l n**

Report Generated When Rerating Is Performed before Billing

Summary reports are created for each account. The following example of a summary report shows that rerating occurred as a result of changing the monthly cycle forward fee from $200 to $20. Rerating took place *before* billing was run, so the difference is shown in the current bill.

```
             PIN_RERATE SUMMARY REPORT
          ========================================

Date Range: 3/1/2007 10:0:0 to 3/2/2007 20:4:49
------------------------------

Account: 0.0.0.1 /account 13640 0
Resource Name   Original Amount      New Amount      Difference
US Dollar       200.000000           20.000000       -180.000000
Total Resources 200.000000           20.000000       -180.000000
------------------------------

++++++++++++++++++++++++++++++++++++
The Total Rerate Impact:
++++++++++++++++++++++++++++++++++++
Resource Name   Original Amount      New Amount      Difference
US Dollar       200.000000           20.000000       -180.000000
Total Resources 200.000000           20.000000       -180.000000
-----------------------------------------------
END OF REPORT
=====================================================================
```

The following is the detailed report for the example above.

```
               PIN_RERATE DETAILED REPORT
            ========================================

Date Range: 3/1/2007 10:0:0 to 3/2/2007 20:4:49

Event: 0.0.0.1 /event/billing/product/fee/cycle/cycle_forward_monthly 219550481834327128
0
Service Type:   /service/ip
Event Time:     3/1/2007 20:4:14
----------
O/N       Resource    Amount       Disct     GL_ID       Tax
Original  US Dollar   -200.000000  0.000      102        0.000000
New       US Dollar    20.000000   0.000      102        0.000000
---------------------------------------
Account: 0.0.0.1 /account 13640 0
Resource Name    Original Amount      New Amount      Difference
```

```
US Dollar        200.000000           20.000000       -180.000000
Total Resources  200.000000           20.000000       -180.000000
-------------------------------

++++++++++++++++++++++++++++++++++++
The Total Rerate Impact:
++++++++++++++++++++++++++++++++++++
Resource Name     Original Amount      New Amount      Difference
US Dollar         200.000000           20.000000       -180.000000
Total Resources   200.000000           20.000000       -180.000000
-----------------------------------------------
END OF REPORT
```

Report Generated When Rerating Is Performed after Billing

The following example of a summary report shows rerating as a result of changing the monthly cycle forward fee from $200 to $20. Rerating took place *after* billing was run, so the difference is posted in the *next* bill.

Adjustments are posted only for cycle forward fees. Since cycle forward fees are charged in advance and since the difference between the old and the new amount in this case is $180, the new amount is calculated as -360. In this case, the reports show only the adjusted amount.

```
                PIN_RERATE SUMMARY REPORT
        ========================================

Date Range: 8/7/2007 10:0:0 to 9/7/2007 20:6:52

-------------------------------
Account: 0.0.0.1 /account 14854 0
Resource Name     Original Amount      New Amount      Difference
US Dollar         0.000000             -360.000000     -360.000000
Total Resources   0.000000             -360.000000     -360.000000
-------------------------------

++++++++++++++++++++++++++++++++++++
The Total Rerate Impact:
++++++++++++++++++++++++++++++++++++
Resource Name      Original Amount      New Amount     Difference
US Dollar          0.000000             -360.000000    -360.000000
Total Resources    0.000000             -360.000000    -360.000000
-----------------------------------------------
END OF REPORT
```

The following is a detailed report of the example above:

```
                PIN_RERATE DETAILED REPORT
        ========================================

Date Range: 8/7/2007 10:0:0 to 9/7/2007 20:6:52


Event:0.0.0.1 /event/billing/adjustment/event 222875404996720238 0
Adjusted From:0.0.0.1 /event/billing/product/fee/cycle/cycle_forward_monthly
222330047229342470 0
Service Type:   /service/ip
Event Time:     9/7/2007 20:6:54
----------
Resource      Amount         Disct     GL_ID
US Dollar     -180.000000    0.000     102
----------
```

```
Event:0.0.0.1 /event/billing/adjustment/event 222875404996721006 0
Adjusted From:  0.0.0.1 /event/billing/product/fee/cycle/cycle_forward_monthly
222875404996719270 0
Service Type:   /service/ip
Event Time:     9/7/2007 20:6:54
----------
Resource          Amount         Disct     GL_ID
US Dollar         -180.000000    0.000     102
-----------------------------------------
Account: 0.0.0.1 /account 14854 0
Resource Name     Original Amount      New Amount      Difference
US Dollar         0.000000             -360.000000     -360.000000
Total Resources   0.000000             -360.000000     -360.000000
-------------------------------

++++++++++++++++++++++++++++++++++++
The Total Rerate Impact:
++++++++++++++++++++++++++++++++++++
Resource Name     Original Amount      New Amount      Difference
US Dollar         0.000000             -360.000000     -360.000000
Total Resources   0.000000             -360.000000     -360.000000
-----------------------------------------------
END OF REPORT
```

Improving pin_rerate Performance

The following parameters can be used with **pin_rerate** to improve performance:

```
- pin_rerate  rerate_children  5
- pin_rerate  rerate_per_step  1000
- pin_rerate  rerate_fetch_size  5000
```

66
Rerating Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) rerating utilities.

load_pin_config_ood_criteria

Use the **load_pin_config_ood_criteria** utility to load out-of-order criteria into the **/config/event_order_criteria** object in the BRM database. You define out-of-order criteria in the **pin_config_ood_criteria.xml** file in *BRM_home***/sys/data/config**.

> **Note:**
>
> To connect to the BRM database, the **load_pin_config_ood_criteria** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.

> **Note:**
>
> The **load_pin_config_ood_criteria** utility overwrites existing out-of-order criteria. If you are updating out-of-order criteria, you cannot load new out-of-order criteria only. You must load complete sets of out-of-order criteria each time you run the **load_pin_config_ood_criteria** utility.

Location

*BRM_home***/bin**

Syntax

```
load_pin_config_ood_criteria [[-d][-v]|[-t] out_of_order_criteria_file.xml ]|[-h]
```

Parameters

-h
Displays online help about the command.

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

-t
Checks the validity of the XML file but does not process any data.

out_of_order_criteria_file
The name and location of the file that defines out-of-order criteria. The default **pin_config_ood_criteria.xml** file is in *BRM_home***/sys/data/config**.
If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

```
load_pin_config_ood_criteria BRM_home/sys/data/config/pin_config_ood_criteria.xml
```

> **Note:**
>
> If you copy the **pin_config_ood_criteria.xml** file to the directory from which you run the **load_pin_config_ood_criteria** utility, you do not have to specify the path or file name. By default, the name of the file is **pin_config_ood_criteria.xml**. You can change this name.

Results

The **load_pin_config_ood_criteria** utility notifies you when it successfully creates the **/config/event_order_criteria** object. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

To verify that out-of-order criteria was loaded, you can display the **/config/event_order_criteria** object by using the Object Browser application in Developer Center, or use the **robj** command with the **testnap** utility.

> **Note:**
>
> You must send a **Reload** semaphore to make new out-of-order criteria available in the Pipeline Manager. The **Reload** semaphore is used by the DAT_PortalConfig data module.

load_pin_rerate_flds

Use this utility to load the following information into the BRM database:

- Load extraction keys-to-event field mappings into the **/config/rerate_flds** object. You define extraction keys and map them to EDR fields by using XML.
- Load balance-impact comparison fields into the **/config/rerate_flds/compare_bi** object. The balance impact fields in this object determine whether the balance impacts of rerating and previous rating are equivalent.

Location

*BRM_home***/ sys/data/config**

Run **load_pin_rerate_flds** from this directory.

Syntax

```
load_pin_rerate_flds -f xml_file_name [-v] [-h]
```

Parameters

-f *xml_file_name*
Specifies the name of the XML file that contains either the extraction keys and field mappings or the balance-impact comparison fields.

> **Note:**
>
> The file you load cannot include both extraction-key mapping and balance-impact comparison fields. You must load the files for these configurations separately.

-verbose
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-verbose** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_rerate_flds** *any_other_parameter* **–v >** *filename***.log**

-help
Displays the syntax and parameters for this utility.

Results

This utility notifies you when it successfully creates the **/config/rerate_flds** object.

If it cannot create the **/config/rerate_flds**, it logs an error in the log file (**default.pinlog**). It creates the log file either in the directory from which the utility was started or in the directory specified in the configuration file.

pin_event_extract

Use this utility to extract events from your database to be rerated by the BRM Pipeline Manager. See "Using Event Extraction Manager" for more information.

You define which events to extract by using the **pin_event_extract.cfg** file located in the *BRM_home***/apps/pin_event_extract** directory.

> **Note:**
>
> To connect to the BRM database, the **pin_event_extract** utility needs a configuration file in the directory from which you run the utility.

Location

BRM_home/**bin**

Syntax

```
pin_event_extract [-f ConfigFileName] [-s] [-r roaming_input_file] [-b BrandName] [-e] [-o] [-h]
```

Parameters

-f *ConfigFileName*
Specifies the name and location of the configuration file that defines which events to extract. For information on how to create the file, look at the sample file (*BRM_home*/**apps/pin_event_extract/pin_event_extract.cfg**).
By default, the utility looks in the current working directory. If your configuration file is located in a different directory, you must specify the entire path for the file. If the **pin_event_extract.cfg** file is located in the current working directory, you can ignore this parameter.

-s
Extracts only events that match the event start time (**event_start_time_stamp**) specified in *ConfigFileName*.
You must use this option together with the **-f** parameter.

-r *roaming_input_file*
Extracts events based on the information provided in *roaming_input_file*, which is generated by the RAP Processing pipeline during Outcollect processing. It contains records that specify the extraction criteria for extracting TAP settlement records from the BRM database.

-b *BrandName*
Extracts events assigned to a specific brand. You can use this option alone or with the **-f** option.

-e
Flags the events for back-out-only rerating. Pipeline Manager backs out the balance impact and does not rerate the event.

-o TRUE | FALSE
Overrides the program lock.

- **-o TRUE** resets the status table to the unused state.
- **-o FALSE** sets the status table to the used state.

The Event Extraction Manager cannot run at the same time as Rated Event Loader or Rerated Event Loader. To prevent this, all three applications use a status table to determine if one of the other applications is running. If one of the applications terminates abnormally and leaves the status table in a locked state, the Event Extraction Manager cannot be started. In this case, use the **-o TRUE** option to reset the status table to an unused state.

> **Note:**
>
> Before using this option, ensure that the Rated Event Loader and Rerated Event Loader are stopped.

-h
Displays the syntax and parameters for this utility.

Results

The **pin_event_extract** utility notifies you when it successfully completes a command.

pin_load_rerate_jobs

This utility is used by the **OODHandler** batch handler to process out-of-order events. It converts rerate-request files generated by the FCT_EventOrder module into flist format and then calls PCM_OP_ACT_USAGE to generate a notification event.

> **Note:**
>
> You must run the utility from a directory that contains both the **pin_rerate_job_info.xsd** file and a **pin.conf** configuration file. By default, these files are located in the *BRM_home***/apps/pin_ood_handler/process** directory.

Location

*BRM_home***/bin**

Syntax

```
pin_load_rerate_jobs [[-d][-v]|[-t] rerate_request_file.xml]|[-h]
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays detailed information as the utility runs.

-t
Checks the validity of the XML file but does not process any data. When you run the file in test mode, it returns either that the XML file is valid or an error message that contains the invalid line number.

***rerate_request_file*.xml**
Specifies the name of the rerate request XML file that is generated by the FCT_EventOrder pipeline module.

-h
Displays online help about the command.

Results

The utility notifies you only if it encounters errors. Look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

pin_rate_change

Use this BRM utility after you change the rates for cycle forward and cycle forward arrears events in the current cycle. This utility triggers the creation of rerating requests that the **pin_rerate** utility uses to recalculate the charges for these events and adjust the balance impacts.

The **pin.conf** file for this utility is in *BRM_home***/apps/pin_rate_change**.

Location

*BRM_home***/bin**

Syntax

```
pin_rate_change [-v] [-d] [-h]
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **pin_rate_change** any_other_parameter **–v >** filename**.log**

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-h
Displays the syntax and parameters for this utility.

Results

The **pin_rate_change** utility notifies you when it successfully creates the **rerating requests**.

If the **pin_rate_change** utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_rel

Loads batches of prerated or rerated event records from Pipeline Manager into the BRM database.

There are two ways to use this utility. When you initially run **pin_rel**, use the command without any options and use the file name as the only command-line parameter. You use the **override** option when the utility has not successfully completed its process and needs to be rerun.

pin_rel records any errors in the **pin_rel** log file (*BRM_home***/apps/pin_rel/rel.pinlog***)*.

Location

BRM_home/**apps/pin_rel**

Syntax

```
pin_rel [-override] event_file_name
```

Parameters

-override
This option starts a new **pin_rel** process. Use this option to restart **pin_rel** when it has abnormally stopped or you have stopped it manually.

> **Note:**
>
> Use this option only when you know there are no other RE Loader processes running.

RE Loader maintains a status of its operations. Because only one RE Loader process can load events into the same database at the same time, the status must indicate the loading process is complete before another process can start loading. If you manually stop **pin_rel**, its status may not reflect its true state. The **-override** parameter overrides the status and permits a new loading process to start, providing one is not already running.

event_file_name
The name of the event file to load, including its extension.

Results

If **pin_rel** is successful, it returns PROCESSED SUCCESSFULLY in the RE Loader log file (*BRM_home*/**apps/pin_rel/rel.pinlog**).

If an error occurs during loading, this utility cancels the loading process. An error is logged in the **rel.pinlog** file, SQL loader errors are logged in a “bad" file (*BRM_home*/**apps/pin_rel/***CDR_file_name*.**bad**), and the records loaded in this session are deleted from the database.

If an error occurs during account updating, the error is logged in the **rel.pinlog** file. Loaded records are not deleted from the database.

pin_rerate

Use this BRM utility to rerate events rated in real-time and events rated in batch by Pipeline Manager, and update account balances with the rerated results.

If you rerate both real-time-rated and pipeline-batch-rated events concurrently, you run **pin_rerate** multiple times: once to create rerating jobs by selecting the accounts and events for rerating, and several times again to process the rerate jobs.

If you do not use Pipeline Manager for batch rating, you select the accounts and events and rerate them by using a single **pin_rerate** command.

> **Note:**
>
> - **pin_rerate** is a multithreaded application.
> - In a multischema environment, you must run **pin_rerate** separately on each database schema.
> - To connect to the BRM database, the **pin_rerate** utility needs a configuration file in the directory from which you run the utility.

Location

BRM_home/**bin**

The **pin.conf** file for this utility is located in *BRM_home*/**apps/pin_rerate**. Run **pin_rerate** from this directory.

Syntax

```
pin_rerate      [-t [ss/mm/hh/] MM/DD/YYYY]
                [-a account POID_id]
                [[-d | -g | -i | -m | -n | -p | -s] input_file]
                [-line subscription_id]
                [-field_name input_file]
                [-r]
                [-reason reason_code]
                [-b [c | e]]
                [-c]
                [-backout]
                [-e  [-a | -s | -p | -n | -d]]
                [-h | help]
                [-purge [-t [ss/mm/hh/] MM/DD/YYYY]]
                [-l [d | s | sd | n]]
                -process jobs [-reason reason_code [,reason_code...]]
                -process queue
                -rerate [-reason reason_code [,reason_code...][-l [d | s | sd | n]]
                -process recycle [-db database_id]
```

Parameter Overview

The parameters specify which accounts and events to rerate and which rerating process to perform.

- To select the accounts and events for rerating, see the following sections:
  - Parameters for Selecting Accounts for Rerating
  - Parameter for Specifying Events for Rerating
- To assign a rerate reason code for rerating, see "Parameter for Assigning a Rerate Reason".
- To specify rerating behavior, such as how to order the events, whether to update the database, and whether to generate reports, see "Parameters Affecting Rerating Behavior".
- To purge rerate jobs, see "Parameter for Purging Rerate Jobs".
- To process rerate jobs and rerate the selected accounts and events, see "Parameters for Processing Rerate Jobs".

Parameters for Selecting Accounts for Rerating

The following parameters are used to select accounts for rerating based on the described event criteria.

All account selection parameters are optional and mutually exclusive, except for the **-t** parameter, which is mandatory when selecting the accounts and events to rerate. If you specify only the time (**-t**) parameter, BRM selects all accounts for rerating in the period defined by the **-t** parameter.

-a [account *POID_id*]
Selects a *single account* for rerating based on the provided account POID.
When you specify an account POID, you can also use the **-c** option to rerate events *without* updating the database.
This parameter is optional.

> **Note:**
>
> You cannot use this parameter with the **-d, -g, -i, -m, -p, -s**, **-line**, or custom parameters.

-d *input_file*
Selects accounts for rerating based on the *bundle*. Accounts that have events associated with the charge offers and discount offers that belong to the bundles specified in *input_file* are selected for rerating.
input_file format: A text file with one bundle name specified on each line. The bundle names are case sensitive.
This parameter is optional.

> **Note:**
>
> If a charge offer is part of multiple bundles, all accounts that purchase the charge offer are selected for rerating even if they did not purchase the bundle that was rerated.

> **Note:**
>
> You cannot use this parameter with the **-a, -g, -i, -m, -n, -p, -s**, **-line**, or custom parameters.

-g *input_file*
Selects accounts for rerating based on one of the following:

- The account and bill unit objects
- The account, bill unit, and balance group objects

Accounts that match any criteria listed in *input_file* are selected for rerating.

- If items in *input_file* specify an account, bill unit, and balance group, only events specific to the balance group are selected.

- If items in *input_file* specify an account and a bill unit only, **pin_rerate** searches for all the balance groups associated with the bill unit and then selects the events specific to those balance groups.

input_file format: A text file containing information for one or more accounts in flist format. Specify an account POID, **/billinfo** POID, and balance group POID in a results array for each account. For example:

```
0 PIN_FLD_RESULTS                  ARRAY [1]
    1   PIN_FLD_ACCOUNT_OBJ          POID [0] $DB_NO /account 12345 0
    1   PIN_FLD_BILLINFO_OBJ         POID [0] $DB_NO /billinfo 34567 0
    1   PIN_FLD_BAL_GRP_OBJ          POID [0] $DB_NO /balance_group 66765 0
0 PIN_FLD_RESULTS                  ARRAY [1]
    1   PIN_FLD_ACCOUNT_OBJ          POID [0] $DB_NO /account 12344 0
    1   PIN_FLD_BILLINFO_OBJ         POID [0] $DB_NO /billinfo 45654 0
    1   PIN_FLD_BAL_GRP_OBJ          POID [0] $DB_NO /balance_group NULL 0
...
```

where:

- PIN_FLD_ACCOUNT_OBJ specifies the POID of the account object.
- PIN_FLD_BILLINFO_OBJ specifies the POID of the bill unit object.
- PIN_FLD_BAL_GRP_OBJ specifies the POID of the balance group object.

To rerate all events for all balance groups for a given account's bill unit, specify a value of NULL as the POID ID for the balance group object, as shown in the second results array in the above example.

This parameter is optional.

> **Note:**
>
> You cannot use this parameter with the **-a**, **-d**, **-i**, **-m**, **-n**, **-p**, **-s**, **-e**, **-line**, or custom parameters.

-i *input_file*
Selects accounts for rerating based on the *discount object*. Accounts that own at least one instance of the discount objects specified in *input_file* are selected for rerating.
input_file format: A text file with one discount name specified on each line. Discount names are case sensitive.
This parameter is optional.

> **Note:**
>
> You cannot use this parameter with the **-a**, **-d**, **-g**, **-m**, **-n**, **-p**, **-s**, **-line**, or custom parameters.

-m *input_file*
Selects a set of accounts for rerating based on the provided account POIDs.
input_file format: A text file containing the accounts to rerate in flist format. Specify each account in a results array. For example:

```
0 PIN_FLD_RESULTS                   ARRAY [1]
    1 PIN_FLD_ACCOUNT_OBJ            POID [0] $DB_NO /account 12345 0
0 PIN_FLD_RESULTS                   ARRAY [2]
    1 PIN_FLD_ACCOUNT_OBJ            POID [0] $DB_NO /account 12333 0
...
```

where PIN_FLD_ACCOUNT_OBJ is the POID of the account object.

> **Note:**
>
> You cannot use this parameter with the **-a**, **-d**, **-g**, **-i**, **-n**, **-p**, or **-s -e**, **-line**, or custom parameters.

-n *input_file*
Selects accounts for rerating based on the *event types*. Accounts that have events of the types specified in *input_file* are selected for rerating.
input_file format: A text file with one event type specified on each line. Event types are case sensitive.
This parameter is optional.

> **Note:**
>
> - You cannot use this parameter with the **-a**, **-d**, **-g**, **-i**, **-m**, **-p**, **-s**, **-line**, or custom parameters.
> - If you use this parameter with the **-r** parameter, all subclasses of the event type specified in the input file are also rerated.
> - When using a custom **pin_rerate** parameter, the **-n** parameter is mandatory if the custom parameter is based on an event field that is present only in an event subclass. In this case, the **-n** parameter input file can contain only one event type.

-p *input_file*
Selects accounts for rerating based on the *charge offer*. Accounts that have events associated with the charge offers specified in *input_file* are selected for rerating.
input_file format: A text file with one charge offer name specified on each line. Charge Offer names are case sensitive.
This parameter is optional.

> **Note:**
>
> You cannot use this parameter with the **-a**, **-d**, **-g**, **-i**, **-m**, **-n**, **-s**, **-line**, or custom parameters.

-s input_file
Selects accounts for rerating based on the *service type*. Accounts that have events associated with the service types specified in *input_file* are selected for rerating.
input_file format: A text file with one service type specified on each line. Service type names are case sensitive.

This parameter is optional.

> **Note:**
>
> You cannot use this parameter with the **-a**, **-d**, **-g**, **-i**, **-m**, **-n**, **-p**, **-line**, or custom parameters.

-t [ss/mm/hh/]MM/DD/YYYY
Selects accounts for rerating based on the event *start time*. All accounts with events that occurred between the start time and the current date are selected for rerating.

> **Note:**
>
> The group of selected accounts can be further restricted by using criteria specified with the **-a**, **-d**, **-g**, **-i**, **-m**, **-n**, **-p**, **-s**, **-line**, or custom parameters.

-line *subscription_id*
Selects accounts for rerating based on a subscription service.
subscription_id specifies the subscription service ID. The subscription service is identified by the PIN_FLD_ALIAS_LIST.PIN_FLD_NAME field, which can specify, for example, the caller ID such as a phone number or the MAC address.
This parameter is optional.

> **Note:**
>
> The group of selected accounts can be further restricted by using criteria specified with the **-a**, **-d**, **-g**, **-i**, **-m**, **-n**, **-p**, **-s**, or custom parameters.

-field_name input_file
field_name is a custom **pin_rerate** parameter you have defined that maps to a specific event field.
input_file is a file that contains the values of the event field that are used to select the accounts for rerating.
Selects accounts based on the custom parameter. Accounts are selected for rerating that have events with the field identified by field_name whose field value is one of those specified in input_file.

> **Note:**
>
> You cannot use this parameter with the **-a**, **-d**, **-g**, **-i**, **-m**, **-p**, **-s**, or **-line** parameters.

> **Note:**
>
> If the custom parameter maps to a field in an **/event** subclass, you must specify the subclass event type by including the **-n** parameter, and the **-n** parameter input file can include only one event type. If you specify more than one event type, an error is returned.

Parameter for Specifying Events for Rerating

Use the following parameter to specify which events to rerate for the selected accounts.

-r
Specifies whether to use *selective rerating*. This applies the account selection criteria parameters (**-a**, **-d**, **-g**, **-i**, **-m**, **-n**, **-p**, **-s**, **-line** or field_name) to the account's events as well and selects events for rerating based on the specified parameter. For example, using **-r** and **-p** input_file in the command line selects accounts based on the charge offers specified in input_file, and then selects and rerates only the account's events associated with those charge offers.

> **Note:**
>
> Do not use selective rerating if your rating scheme includes credit limits or balance element-based tiers. These schemes require that all events related to an account are rerated to ensure accurate rerating.
> Do not use selective rerating if deferred taxation is used for taxing events during rerating.
> Use caution when specifying selective rerating. Account balances can be impacted by different types of events, and the order of the balance impacts is important to accurately apply discounts and consume balance elements. It is typically safer to rerate all of an account's events.

This parameter is optional and can be used with any other parameter. The value for this parameter can be specified as either **0 (disable)** or **1 (enable)**.

> **Note:**
>
> If no value is specified for this parameter, the default behavior is to consider the value as **1**.

Parameter for Assigning a Rerate Reason

-reason *reason_code*
Use with other selection parameters to assign a rerate reason code to the jobs created for the selected accounts.
Reason code format: Any integer except **1** (**1** is a reserved default value for pipeline-generated rerating requests).
If you create rerate jobs through BRM automatic rerating, rerate reasons are hard coded by the opcodes that handle those rerating scenarios. Be sure your rerate reasons are unique to process rerate jobs associated with them during separate rerating executions.

To process rerate jobs according to a rerate reason, see "Parameters for Processing Rerate Jobs".

Parameters Affecting Rerating Behavior

The following are additional, optional parameters that affect rerating behavior and the utility's output.

-b [c | e]
Specifies the order in which *batch* events are rated. The **c** option rerates events in the order that they were created in the BRM database. The **e** option rerates events based on the time when the event actually occurred. The default is **e**.

> **Note:**
>
> Due to real time account balance impacts and credit limit monitoring, the order in which the batch events are processed may result in different rating results.

This parameter is optional.

-c
Specifies calculation of rerating only. Use this parameter to log the rerating result of an account into a report file without updating the database. This option can be used for only one account, so it works only when using the **-a** parameter.
This parameter is optional.

-backout
Specifies back-out-only rerating, which backs out the balance impacts of rating without rerating events.
This parameter is optional and can be used with any other parameter.

> **Note:**
>
> When choosing the events to back out, ensure that you do not select events that could imbalance your general ledger, such as events for which fees have already been paid and usage events that should be rerated. Typically, back-out-only rerating is performed only on usage events where rating should not have occurred.

-e [-a | -s | -p | -n | -d]
Returns an estimate of how many events might be affected by rerating based on which accounts are being rerated. Options:
-e -a: A single account
-e -s: Accounts with a specific service type
-e -p: Accounts with specific charge offers
-e -n: Accounts with specific event types
-e -d: Accounts with specific bundles

> **Note:**
>
> If you do not specify one of these options when using the **-e** parameter, the utility returns **0**.

The **pin_rerate** utility prints the output of this option on the screen or to the **pin_rerate.pinlog** file in **$PIN_LOG/pin_rerate**.
This parameter is optional.

> **Note:**
>
> You cannot use this parameter with the **-m** or **-g** parameters.

-h | help
Displays the syntax and parameters for this utility.

-l [d | s | sd | n]
Specifies whether to generate a report. Options:
-ld: Specifies a detailed report (**pin_rerate.detail**).
-ls: Specifies a summary report (**pin_rerate.summary**).
-lsd: Specifies both detailed and summary reports. This is the default.
-ln: Specifies no report.

> **Note:**
>
> When Pipeline Manager is running, you can specify these report options at rerate time (when you use the **-rerate** parameter). You cannot specify them when you create the rerate jobs.

Parameters for Processing Rerate Jobs

You use different parameters to process rerate jobs depending on whether you rerate real-time-rated and pipeline-batch-rated events concurrently or, if you do not use Pipeline Manager for batch rating, rerate only real-time-rated events.

- Processing Rerate Jobs When Rerating Real-Time and Pipeline Rated Events Concurrently
- Processing Rerate Jobs When You Do Not Use Pipeline Manager

Processing Rerate Jobs When Rerating Real-Time and Pipeline Rated Events Concurrently

To process rerate jobs when rerating real-time and pipeline rated events concurrently, you run the following commands in the order shown.

```
pin_rerate -process jobs [-reason comma_separated_reason_codes]
pin_rerate -process queue
pin_rerate -rerate [-l  [d | s | sd | n] ]
pin_rerate -process queue
pin_rerate -process recycle
```

> **Note:**
>
> You must run **pin_rerate** with all of the above parameters to complete the rerating process.

These commands process rerate jobs that were created both manually, by running **pin_rerate** to select accounts for rerating (see "Parameters for Selecting Accounts for Rerating"), and automatically, by BRM automatic rerating features.

The parameters specify the following:

-process jobs [-reason comma_separated_reason_codes]
Searches for all rerate jobs with status equal to NEW, and notifies Pipeline Manager to lock the accounts in the rerate jobs. Then updates the status of the rerate jobs to WAITING_ACCOUNT_LOCK.
If you use the **-reason** parameter, performs the same actions but only for the rerate jobs associated with the specified reason codes, where:

comma_separated_reason_codes
specifies the codes you assigned as the rerate reason for the jobs you want to process.

> **Note:**
>
> If you do not specify a rerate reason code, all rerate jobs in the rerate queue are processed.

-process queue
Processes acknowledgment events from Pipeline Manager and updates rerate job status as follows:

- If acknowledging notification to suspend EDRs, it updates the status to ACCOUNT_LOCKED.
- If acknowledging notification to resume processing EDRs, it updates the status to READY_TO_RECYCLE.

This parameter is used twice: before and after the **-rerate** parameter.

-rerate [-l [d | s | sd | n]]
Rerates events for the accounts referenced in the rerate jobs, and updates the rerate job status to RERATED.

-l
Specifies to generate a rerating report. Options are:
-ld: Generates a detailed report (**pin_rerate.detail**).
-ls: Generates a summary report (**pin_rerate.summary**).
-lsd: Generates both detailed and summary reports. This is the default.
-ln: Generates no report.

-process recycle [-db *database_id*]
Recycles the EDRs suspended during the rerating process, and updates the rerate job status to COMPLETE.

-**db** *database_id*: Specifies the ID of the database schema that contains the suspended EDRs in the format *x.x.x.x* (for example, 0.0.0.2). Use this only if the suspended EDRs are not stored in the current BRM schema.

> **Note:**
>
> If you do not specify the **-db** *database_id* parameter, the suspended EDRs are picked up from the current schema.

Processing Rerate Jobs When You Do Not Use Pipeline Manager

If you do not use Pipeline Manager for batch rating, rerate jobs for real-time-rated events are processed as follows:

- When you run **pin_rerate** with parameters that specify the accounts and events to rerate, **pin_rerate** creates the rerate jobs for those accounts are then immediately rerates their events.
- Rerate jobs that already exist (such as those created by automatic rerating) are processed with the following parameter:

-rerate [-reason comma_separated_reason_codes]
Rerates events for the accounts referenced in the rerate jobs, and updates the rerate job status to COMPLETE.
If you use the **-reason** parameter, performs the same actions but only for the rerate jobs associated with the specified reason codes.
comma_separated_reason_codes
Specifies the codes you assigned as the rerate reason for the jobs you want to process.

> **Note:**
>
> If you do not specify a rerate reason code, all rerate jobs in the rerate queue are processed.

Parameter for Purging Rerate Jobs

Use the following parameter to purge rerate jobs.

-purge [-t [*ss/mm/hh/*] *MM/DD/YYYY*]
Purges rerate jobs that have a rerate job status of COMPLETE or UNSUCCESSFUL.
[**-t** [*ss/mm/hh/*] *MM/DD/YYYY*]
Specifies to purge rerate jobs that have had their rerate job status set to COMPLETE or UNSUCCESSFUL before the time specified.
If no time is specified, all rerate jobs that have a rerate job status of COMPLETE or UNSUCCESSFUL are purged up to the current date.

Results

If the **pin_rerate** utility does not notify you that it was successful, look in the utility log file (**pin_rerate.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

If rerating fails, the utility creates a report that includes the account numbers and start times for failed rerates. The report file name is **pin_rerate.status_report**, and is in the directory from where you ran the utility.

Part VIII

Pipeline Manager System Administration

This part describes how to run, maintain, monitor, and troubleshoot Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. It contains the following chapters:

- Pipeline Manager System Architecture
- Configuring Pipeline Manager
- Starting and Stopping Pipeline Manager
- Monitoring Pipeline Manager
- Optimizing Pipeline Manager Performance
- Migrating Accounts with the Pipeline Manager Running
- Pipeline Manager Error Messages

67
Pipeline Manager System Architecture

This chapter describes Oracle Billing and Revenue Management Pipeline Manager system architecture.

About the Pipeline Manager System Architecture

Pipeline Manager is used for rating and discounting events in batch and real-time.

> **Note:**
>
> BRM pipeline batch rating engine and BRM real-time pipeline used for advanced discounting do not support SSL/TLS connections. Therefore, the communication between BRM CM and BRM real-time pipeline is not encrypted.

The Pipeline Manager system architecture consists of:

- The pipeline framework that controls the Pipeline Manager system functions.
- The pipelines that the framework runs, which perform rating and discounting.
- The data pool that provides data in memory, used for rating and discounting.
- The Pipeline Manager database that stores data used for rating and discounting.

Figure 67-1 shows how a billable event is rated in batch by Pipeline Manager and recorded in the BRM database. In this case:

1. Pipeline Manager rates event data from CDR files.
2. Rated Event (RE) Loader loads rated events into the BRM database.
3. Account balances are updated.

Figure 67-1 Billable Event Rating by Pipeline Manager and Storage in BRM Database

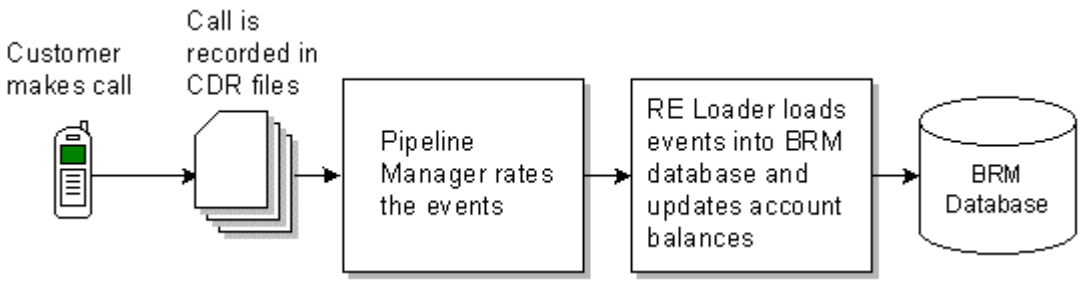
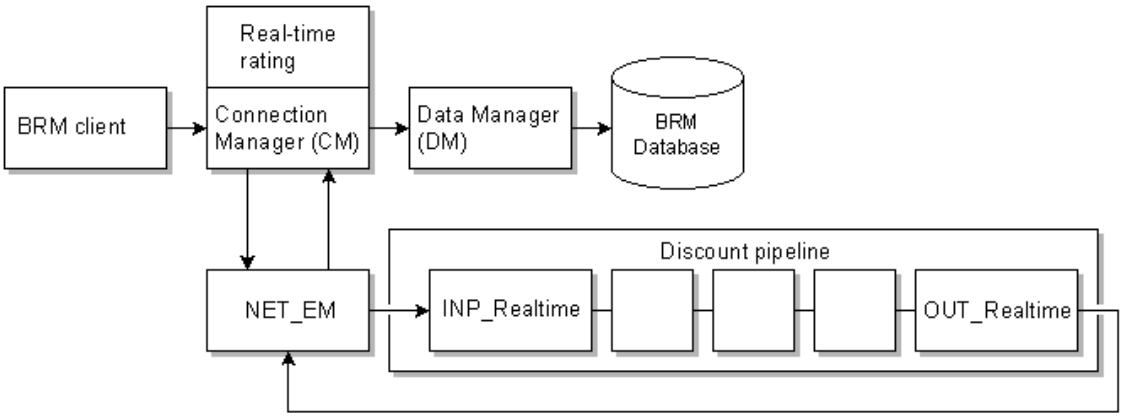


Figure 67-2 shows how real-time discounting works.

Figure 67-2 Real-Time Discounting



In this case:

1. BRM sends an event to the pipeline for real-time discounting.
2. The NET_EM module sends the event to the pipeline.
3. Pipeline Manager returns the discounted amount.
4. Account balances are updated in the BRM database.

About the Pipeline System Components

When you configure an instance of the Pipeline Manager, you configure a set of system components and one or more pipelines. The system components are:

- Controller. See "About the Controller".
- EDR Factory. See "About the EDR Factory".
- Transaction ID Controller. See "About the Transaction ID Controller".
- Sequencer. See "About the Sequencer".
- Event Handler. See "About the Event Handler".

About the Controller

The Controller manages and monitors the entire Pipeline Manager instance. The Controller performs these functions:

- Starts and stops a Pipeline Manager instance.
- Initiates and coordinates different threads.
- Checks for new semaphore file entries.
- Generates a log message table that is used by the LOG module to create the process log file, the pipeline log files, and the stream log file.

You configure the Controller by using the registry file.

About the EDR Factory

The EDR Factory is a mandatory pipeline component that generates and allocates memory to EDR containers in a single pipeline.

When a transaction starts, the EDR Factory:

1. Allocates memory for each container.
2. Generates an EDR container for each piece of the input stream, including one for the header, one for each EDR, and one for the trailer, by using the container description file.
3. After the pipeline writes information to the output file, the EDR Factory empties the container and releases the cache. The EDR Factory can then reuse the memory for new containers.

You configure the EDR Factory by using the **EDRFactory** section of the registry file.

About the Transaction ID Controller

The Transaction ID Controller generates unique IDs for all open transactions in your pipelines. An instance of Pipeline Manager contains only one Transaction ID Controller.

The Transaction ID Controller performs these functions:

- Stores blocks of transaction IDs in cache. The Transaction ID Controller issues IDs to TAMs directly from cache.
- Uses the transaction state file or table to track ID numbers.
- Assigns ID numbers to transactions.

You configure the Transaction ID Controller by using the **TransactionIDController** section of the registry file.

About the Sequencer

The BRM Sequencer is an optional Pipeline Manager component that performs one of these functions:

- *Sequence checking*, which ensures that a CDR file is not processed more than once by keeping track of each CDR file's unique sequence number. A sequence check also logs gaps in sequence numbers.
- *Sequence generation*, which generates sequence numbers for output files. This functionality is used when CDR input files do not have sequence numbers and when pipelines split CDR input files into multiple output files.

> **Note:**
>
> Sequence generation is not required when there is a one-to-one correspondence between input and output files. In this case, sequence numbers can be passed through to the output file.

Each pipeline can be configured to use one or more Sequencers. You configure your Sequencers by using the **SequencerPool** registry entries, and you assign Sequencers to pipelines by using the **Output** registry entries.

About the Event Handler

The Event Handler is an optional pipeline framework component that starts external programs when triggered by internal events. For example, you can configure the Event Handler to launch a script that moves event data record (EDR) output files to a specific directory whenever the output module finishes processing them.

An instance of the Pipeline Manager uses only one Event Handler, which monitors the events for all pipelines in your system. Each registered module in your system automatically sends events to the Event Handler. You define which of these events trigger external programs by using the **ifw.EventHandler** section of the registry file.

When the Event Handler receives an event from a registered module, it:

1. Checks to see if the event is mapped to an action.
2. Performs one of the following:
   - Starts the associated program or script.
   - If no action is mapped, ignores the event.
3. Queues any events it receives while the external program is running.
4. Waits for the external program to terminate.

About the Data Pool

The data pool is a set of modules that store data used by all the pipelines in a single Pipeline Manager instance. Data modules are named with the prefix “DAT", for example, DAT_AccountBatch.

Data modules get their data from the Pipeline Manager database and from the BRM database at startup. As data changes in the BRM system, the data is updated in the data pool.

About Pipelines

A single Pipeline Manager instance runs one or more pipelines. Each pipeline includes the following components:

- The *Pipeline Controller*, which you use to manage the pipeline.
- The *input module* reads data from the input stream, converts CDR files into the internal EDR input format, and performs error checking on the input stream.
- *Function modules* perform all rating tasks and EDR management tasks for a pipeline. Function modules process the data in the EDRs. Each function module performs a specific task, for example, checking for duplicate EDRs or calculating zones.

  Function modules do not store any data; instead they get data from data modules. For example, to rate an event, the FCT_MainRating module gets pricing data from the DAT_PriceModel module.

  Function modules have two dependencies:

  – Some modules require previous processing by other modules.

  – Some modules get data from data modules.
- The *output modules* convert internal EDRs to output format and write the data to the output streams.

- The *log module*, which you use to generate and manage your process, pipeline, and stream log files.

About Using Multiple Pipelines

You create multiple pipelines to do the following:

- Maximize performance and balance system loads. For example, you can create multiple pipelines to handle multiple input streams.
- Manage different types of processing. For example, you can create separate pipelines for zoning, rating, and preprocessing. In this case, you can use the output of one pipeline as the input for another pipeline, or pipelines can run in parallel. To improve performance, aggregation is typically performed in a separate pipeline.

When you create multiple pipelines, they run in parallel in a single Pipeline Manager instance. You configure all pipelines in the same registry file. Each pipeline has its own input and output configuration, EDR Factory, Transaction Manager, and set of function modules. However, all pipelines share the same set of data modules.

You can also use a pipeline to route EDRs to different Pipeline Manager instances. For example, when you use multiple database schemas, you use the FCT_AccountRouter module to send EDRs to separate instances of Pipeline Manager.

About the Pipeline Controller

The Pipeline Controller manages all processes for one pipeline.

The Pipeline Controller performs the following functions:

- Starts and stops the pipeline.
- Initiates and coordinates the pipeline's threads.
- Defines the valid country codes and international phone prefixes for the pipeline. The pipeline's function modules retrieve this information during processing.
- Manages pipeline input and output.

You configure the Pipeline Controller by using the **Pipelines** section of the registry file.

About Thread Handling

You can configure each pipeline to run with *multithreaded* processing or *single-threaded* processing. By default, each pipeline is configured for multithreaded processing.

You select single-threaded or multithreaded mode to optimize performance.

About the Pipeline Manager Database

The Pipeline Manager database stores business configuration data, such as pricings and charges. Pipeline Manager accesses this information when you first start Pipeline Manager or when you force a database reconnection. Pipeline Manager then stores a copy of your pricing and rating data in your data modules.

Pipeline Manager modules connect to the Pipeline Manager database through the Database Connect module (DBC).

About Configuring Pipeline Manager

To configure Pipeline Manager, you use the following files to manage the Controller:

- *Registry files*, which you use to configure a Pipeline Manager instance at system startup.
- *Semaphore files*, which you use to configure and control pipelines during run time.

You can also use the **pin_ctl** utility to start and stop Pipeline Manager.

68
Configuring Pipeline Manager

This chapter describes how to manage Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager framework components and pipelines.

About Configuring Pipeline Manager

To configure Pipeline Manager, you use the following files:

- *Registry files*, which you use to configure a Pipeline Manager instance at system startup, or the BRM wallet. The BRM Installer stores the configuration data provided during the Pipeline Manager installation in the BRM wallet. You can store or change the passwords for pipleline modules using the **pin_crypt_app** utility. See "Storing Passwords for Pipeline Modules in Server Wallet".
- *Semaphore files*, which you use to configure and control pipelines during runtime. See "Using Semaphore Files to Control Pipeline Manager".

You can also use the **pin_ctl** utility to start and stop Pipeline Manager.

Encrypting Pipeline Manager Passwords

You can encrypt passwords for the Pipeline Manager database.

To encrypt the Pipeline Manager database password:

1. Run the **pin_crypt_app** utility to encrypt the Pipeline Manager password.
2. Add the password to the **DataPool** section of the Pipeline Manager registry file.

See "About Encrypting Passwords" for more information.

Storing Passwords for Pipeline Modules in Server Wallet

To store passwords for Pipeline modules in a server wallet:

1. Go to the *BRM_home***/bin** directory.
2. Run this command:

   ```
   pin_crypt_app -setconf -wallet walletLocation -program programName -parameter
   configentry  -pwd
   ```

   where:

   - *walletLocation* is the path to the client wallet.
   - *programName* is the name of the program that is storing the configuration entry.
   - *configEntry* is one of the following:
     - ifw.DataPool.Login.Module.PassWord
     - ifw.DataPool.LoginInfranet.Module.PassWord
     - ifw.DataPool.LoginQueue.Module.PassWord

3. At the prompt, enter the server wallet password.
4. At the prompt, enter the password that you want to store.

> **Note:**
>
> If you want to store the encrypted password, enter the encrypted password at the prompt.

The passwords are stored in the server wallet.

About Configuring Pipelines

Pipelines perform the Pipeline Manager functions, such as rating and zoning.

You configure pipelines in the **ifw.Pipelines** registry section. For example, a Pipeline Manager configuration with multiple pipelines looks like this:

```
ifw
{
...
    Pipelines
    {
        PipelineName
        {
            Input
            Functions
            Output
        }
        PipelineName
        {
            ...
        }
    }
...
```

You can use any name you want to identify pipelines. You use that name in many places to point to the pipeline, so it should identify the function of the pipeline.

For each pipeline, you configure a pipeline controller. This section configures pipeline-specific configurations, such as threads, log files, the EDR Factory, and the **ifw.Pipelines.DataDescription** section.

- Using Events to Start External Programs
- About Pipeline Manager Transactions
- About Pipeline Manager Log Files

In addition, for each pipeline controller, you configure:

- An input section.
- An **ifw.Pipelines.Functions** section. This section configures the function modules in the pipeline. The modules are run in the order that they are configured in this section. See "About Configuring Function Modules".
- An output section.

The registry subsections in a pipeline are listed in Table 68-1:

Table 68-1 Pipeline Registry Subsections

Registry Entry	Description	Required
ifw.Pipelines	Section that configures your individual pipelines.	Yes
ifw.Pipelines.*PipelineName*	Section that configures a single pipeline.	Yes
ifw.Pipelines.*PipelineName***.Input**	Section that configures a pipeline's input module.	Yes
ifw.Pipelines.*PipelineName***.Functions**	Section that configures a pipeline's function modules. For information about the function module entries, see "About Configuring Function Modules".	Yes
ifw.Pipelines.*PipelineName***.Output**	Section that configures a pipeline's output module.	Yes

About Configuring Function Modules

You configure function modules in the **ifw.Pipelines.Functions** section.

The **ifw.Pipelines.Functions** section uses this hierarchy:

```
ifw
{
...
  Pipelines
  {
    PipelineName
    {
      Input
      ...
      Functions
      {
          Function_pool_name
          {
               FunctionPool
               {
                    Module_identifier
                    {
                        ModuleName = Module_executable
                        Module
                        {
                             Entry = value
...
```

The entries listed in Table 68-2 are a combination of required text and text that you define.

Table 68-2 Pipeline Registry Functions Section Entries

Entry	Description
Functions	Section name. You must use **Functions**.
Function_pool_name	The name of the function pool. You define this name.
FunctionPool	Section name. You must use **FunctionPool**.

Table 68-2 (Cont.) Pipeline Registry Functions Section Entries

Entry	Description
Module_identifier	The descriptive module identifier. For example, the module identifier for FCT_Account in the sample registry is **CustomerSearch**. You define these names. They are often referenced by other modules; for example, to connect to the DAT_AccountBatch module, the FCT_Account module points to **CustomerData**.
ModuleName = *Module_executable*	**ModuleName** is the entry. You must use **ModuleName**. *Module_executable* is the name of the module; for example, FCT_Account. This name is case-sensitive and must be spelled correctly; for example, you must use FCT_Account, not FCT_account or FCT_ACCOUNT. You can find the exact spelling and capitalization by looking at the executable name in the *pipeline_home***/lib** directory.
Module	Section name. You must use **Module**.
Entry = *value*	These are the registry entries, for example: **Active = True**

This example shows a sample hierarchy. This sample does the following:

- Creates a function pool named *PreProcessing*.
- Runs the FCT_IRules module, using the identifier *PipelineSplit*.

```
Functions
{
    PreProcessing
    {
        FunctionPool
        {
            PipelineSplit
            {
                ModuleName = FCT_IRules
                Module
                {
                    Active = True
```

About iScripts and iRules

iScripts and iRules perform processing tasks similar to function modules. They are run by the FCT_iScript and FCT_iRules modules. In addition to the iScripts and iRules provided by BRM, you can create your own iScripts and iRules.

About Configuring iScripts

To run iScripts, you use the FCT_IScript module.

The registry section for the FCT_IScript module includes the script to run, for example:

```
ApplyTaxIScript
{
    ModuleName = FCT_IScript
    Module
    {
```

```
        Active = True
        Source = File
        Scripts
        {
            ApplyTaxIScript
            {
                FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ApplyTax.isc
            }
        }
    }
}
```

You can provide registry parameters to use in the iScript. This example provides the iScript with a G/L code:

```
Scripts
{
    ConsolidatedCPIScript
    {
        FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ConsolidatedCP.isc
        GL_CODE = 1514
    }
}
```

About Configuring iRules

To run iRules, you use the FCT_IRules modules.

To configure the FCT_IRules module, provide a connection to the Pipeline Manager database. The FCT_IRules module runs the rules that apply to the conditions in the pipeline. If a condition in a rule item matches the current EDR container, the evaluation stops and the script associated with the rule item is run for the current EDR container.

This example shows a typical FCT_IRules registry section:

```
PipelineSplit
{
    ModuleName = FCT_IRules
    Module
    {
        Active = TRUE
        Source = Database
        DataConnection = integrate.DataPool.DataConnection
        Rules
        {
        }
    }
}
```

You can use the **Rules** entry to specify a specific script to run:

```
Rules
{
   TAP3_VAL
}
```

Configuring Multiple Instances of a Pipeline

To simplify the configuration of multiple pipelines, use the **ifw.Pipelines.Instances** subsection. Pipeline Manager reads the required number of instances for a given pipeline and instantiates each of them accordingly.

> **Note:**
>
> The **ifw.Pipelines.Instances** subsection creates multiple instances of pipelines. To create multiple instances of sequencers, output streams, or system brands for multiple roaming partners, use the Instances module. See "About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands" for more information.

For example, this subsection configures ten instances of the authorization pipeline:

```
ifw
{
...
    Pipelines
    {
        Instances
        {
            AuthPipeline
            {
                NumberOfInstances = 10
                InstanceSpecificRegistries
                {
                Entry1 = TransactionManager.BinaryLogFileName
                Entry2 = PipelineLog.Module.ITO.FileName
                …
                }
            }
        }
    }
}
```

To specify instance-specific registry entries, you add the entries in the **ifw.Pipelines.Instances.***Pipeline_Name*.**InstanceSpecificRegistries** section.

The pipeline generates the instance-specific log file names by adding the instance ID to the base pipeline file names.

For example, if the base pipeline file name for the TransactionManager log file is **binaryLogFile_RT_GPRS.dat**, then the instance-specific files generated are **binaryLogFile_RT_GPRS.dat0**, **binaryLogFile_RT_GPRS.dat1**, and **binaryLogFile_RT_GPRS.dat2**.

> **Note:**
>
> If instance-specific entries are not specified, the pipeline uses the base pipeline configurations.

About Configuring Multiple Instances of Sequencers, Output Streams, or System Brands

To manage multiple roaming partners, you can use the Instances module to configure multiple instances of sequencers, output streams, or system brands. You configure the Instances

module by adding the **ifw.Instances** registry section in the roaming registry file (*pipeline_home*/**conf/roaming.reg**).

> **Note:**
>
> To create multiple instances of pipelines, use the **ifw.Pipelines.Instances** subsection. See "Configuring Multiple Instances of a Pipeline" for more information.

The Instances module configures multiple instances of sequencers, output streams, or system brands using template sections or entries in the roaming registry file. Instead of creating multiple sections of entries, you use the single section or entry templates in the roaming registry file. When the pipeline runs, data for each roaming partner is inserted into the templates, effectively instantiating multiple registry sections or entries. For example, if there are two roaming partners, OPRT1 and OPRT2, the template is instantiated into two sections of entries in the pipeline.

To identify which roaming partners to use with the template, the Instances module reads the roaming configuration data file generated by the **RoamingConfigGen64** utility. This file includes data for each of the roaming partners. For example, the data can include the sequencing information, output information, and so on.

You use the **SequencerPool** or **OUT_GenericStream** template section or the **SystemBrands** template entry in the roaming registry file to configure multiple sequencers, output streams, or system brands. These template sections or entries contain the variables that must be changed in each new instance of the **SequencerPool** or **OUT_GenericStream** section or the **SystemBrands** entry instantiated in the pipeline.

The following example shows the **SequencerPool** template section:

```
SequencerPool
{
SEQ_GEN_TAPOUT_XXX
{
  Source = Database
  Controller
  {
   SequencerType = Generation
   ReuseGap = True
   SequenceLength = 5
   DatabaseConnection = ifw.DataPool.Login
  }
```

where **XXX** is the visiting network operator code that must be changed in each new instance of the **SequencerPool** section; for example, OPRT1, OPRT2, and so on.

Use the Instances module with the **RoamingConfigGen64** utility. The **RoamingConfigGen64** utility collects the roaming partner information from the Pipeline Manager database and creates the roaming configuration data file. The Instances module uses the values in the roaming configuration data file to replace the variables in each instance of the **SequencerPool** or **OUT_GenericStream** section or the **SystemBrands** entry instantiated in the pipeline.

When you run the **RoamingConfigGen64** utility, you specify a home network operator code. The utility searches the Pipeline Manager database to find the VPLMNs associated with that home network operator. For example, if the home network operator has two VPLMNs, a record for each of them is created in the roaming configuration data file.

The following example shows the roaming configuration data file generated by the **RoamingConfigGen64** utility:

```
# Column Headers
###############################################################################
###############################################################################
VPLMN|TAPOUT_SEQUENCER|NRTRDEOUT_SEQUENCER|TAPOUT_STREAM|NRTRDEOUT_STREAM|
TAPOUT_PATH|NRTRDEOUT_PATH|TAPOUT_PREFIX|NRTRDEOUT_PREFIX|TMP_PREFIX|
TMP_DATA_PREFIX###############################################################################
###############################################################################################
##OPRT1|SEQ_GEN_TAPOUT_OPRT1|SEQ_GEN_NRTRDEOUT_OPRT1|TAPOutput_OPRT1|
NRTRDEOutput_OPRT1|./data/outcollect/tapout/oprt1|./data/outcollect/nrtrdeout/oprt1|
CDEUR01OPRT1|NREUR01OPRT1|temptest_oprt1|temp.oprt1.tmp.|42|5|OUT_DevNullOPRT2|
SEQ_GEN_TAPOUT_OPRT2|SEQ_GEN_NRTRDEOUT_OPRT2|TAPOutput_OPRT2|NRTRDEOutput_OPRT2|./data/
outcollect/tapout/oprt2|./data/outcollect/nrtrdeout/oprt2|CDEUR01OPRT2|NREUR01OPRT2|
temptest_oprt2|temp.oprt2.tmp.|42|5|OUT_GenericStream
```

The following example shows the entries in the **ifw.Instances** registry section to configure multiple instances of sequencers:

```
{
 ifw
 {
   Instances
   {
   SEQ_GEN_TAPOUT
   {
     BlockName  = SequencerPool.SEQ_GEN_TAPOUT_XXX
     DataFile = ./RoamingPartnerConf.dat
     InstanceSpecificEntries
     {
         ModifyBlockName
         {
           Instance = [BlockName]
           UseColumn = TAPOUT_SEQUENCER
         }
      }
     }
```

The following example shows the two instances of sequencers instantiated in the pipeline, based on the entries in the **ifw.Instances** registry section, using the TAPOUT_SEQUENCER values in the data file:

```
SequencerPool
{
SEQ_GEN_TAPOUT_OPRT1
{
     Source = Database
     Controller
     {
        SequencerType = Generation
        ReuseGap = True
        SequenceLength = 5
        DatabaseConnection = ifw.DataPool.Login
      }
}
SEQ_GEN_TAPOUT_OPRT2
{
     Source = Database
     Controller
     {
        SequencerType = Generation
```

```
        ReuseGap = True
        SequenceLength = 5
        DatabaseConnection = ifw.DataPool.Login
      }
}
```

See "Configuring Multiple Instances of Sequencers, Output Streams, or System Brands" for instructions.

Configuring Multiple Instances of Sequencers, Output Streams, or System Brands

To configure multiple instances of sequencers, output streams, or system brands:

1. Create the roaming configuration data file by running the following command:

RoamingConfigGen64 -l *database_access_library* **-s** *server_name* [**-d** *database_name*] **-c** *operator_code* [**-o** *output_path*] [**-b** *base_path*]

where:

- *database_access_library* is the database access library.
- *server_name* specifies the name of the host machine running the Pipeline Manager database.
- *database_name* specifies the database name of the Pipeline Manager database. The default is an empty string (' ').
- *operator_code* specifies the home network operator code. The default is **PORTL**.
- *output_path* specifies the output path for the data file generated by the **RoamingConfigGen64** utility. By default, the data file is saved in the *pipeline_home***/conf/** directory.
- *base_path* specifies the base path to the directory for Transferred Account Procedure (TAP) and Near Real Time Roaming Data Exchange (NRTRDE) output files. The default path is *pipeline_home***/data/outcollect/**

For example:

```
RoamingConfigGen64 -l liboci10g6312d.so -s $ORACLE_SID -d ' ' -c EUR01 -
o pipeline_home/conf/ -b pipeline_home/data/outcollect/
```

2. Open the roaming registry file (*pipeline_home***/conf/roaming.reg**) file in a text editor.
3. Ensure that the **SequencerPool** or **OUT_GenericStream** template section or the **SystemBrands** template entry exists in the roaming registry file.

   If the template for the roaming registry section or entry you want to instantiate does not exist, create a template for that registry section or entry in the file.

   The following example shows the **SequencerPool** template section:

```
SequencerPool
{
SEQ_GEN_TAPOUT_XXX
{
  Source = Database
  Controller
  {
   SequencerType = Generation
   ReuseGap = True
   SequenceLength = 5
```

```
   DatabaseConnection = ifw.DataPool.Login
  }
```

4. Add the instance-specific entries in the **ifw.Instances.***InstantiationName***.InstanceSpecificEntries** subsection. If the **ifw.Instances** registry section does not exist, you must add the section in the file.

   The **ifw.Instances** registry section uses the following hierarchy:

```
Instances
{
   InstantiationName
    {
        BlockName  =TemplatePath
        DataFile =DataFilePath
      InstanceSpecificEntries
        {
            InstanceChangeName
            {
                Instance = InstanceValue
                UseColumn = ColumnName
                Mode = ModeValue
            }
        }
}
```

   where:

   - *InstantiationName* is the descriptive name of the instantiation; for example, **SEQ_GEN_TAPOUT**.
   - *TemplatePath* is the template section or entry in the roaming registry file that is used to instantiate multiple registry sections or entries. For example, **SequencerPool.SEQ_GEN_TAPOUT_XXX**
   - *DataFilePath* is the path to the data file generated by the **RoamingConfigGen64** utility; for example, *pipeline_home***/conf/RoamingPartnerConf.dat**.
   - *InstanceChangeName* is the descriptive name of the change required in each instance; for example, ModifyBlockName.
   - *InstanceValue* specifies whether to change the section name, entry name, or the value of the entry in each new instance created.

     The valid values are:

     – **[BlockName]** specifies that the section name or entry name must be changed in each new instance.

     – **[BlockValue]** specifies that the value of the entry must be changed in each new instance.

     – *RegistryEntry* specifies the entry in the template section for which the value must be changed in each new instance; for example, **Module.Recipient**.

   - *ColumnName* is the column in the data file generated by the **RoamingConfigGen64** utility that is used to change the section name, entry name, or the value of the entry in each instance according to the change mode. For example, TAPOUT_SEQUENCER.
   - *ModeValue* is the mode of changing (such as **REPLACE**) the section name, entry name, or the value of the entry in each instance using the column values in the data file generated by the **RoamingConfigGen64** utility.

5. Save and close the file.
6. Stop and restart Pipeline Manager.

Configuring the Data Pool

To configure data modules, you configure the **ifw.DataPool** registry subsection. This subsection uses the following hierarchy:

```
DataPool
{
    Module_identifier
    {
        ModuleName = Module_executable
        Module
        {
            Entry = value
```

The entries listed in Table 68-3 are a combination of required text and text that you define.

Table 68-3 Pipeline Registry DataPool Section Entries

Entry	Description
DataPool	Section name. You must use **DataPool**.
Module_identifier	The descriptive module identifier. For example, in the sample registry, the module identifier for DAT_AccountBatch is **CustomerData**. You define these names. They are often referenced by other modules; for example, to connect to the DAT_AccountBatch module, the FCT_Account module points to **CustomerData**.
ModuleName = *Module_executable*	**ModuleName** is the entry. You must use **ModuleName**. *Module_executable* is the name of the module; for example, DAT_AccountBatch. This name is case-sensitive and must be spelled correctly; for example, you must use DAT_AccountBatch, not DAT_Accountbatch or DAT_Account_Batch. You can find the exact spelling and capitalization by looking at the executable name in the *pipeline_home***/lib** directory.
Module	Section name. You must use **Module**.
Entry = *value*	These are the registry entries; for example: **Active = True**

This example shows a sample hierarchy:

```
DataPool
{
    CustomerData
    {
        ModuleName = DAT_AccountBatch
        Module
        {
            IntegrateConnection = ifw.DataPool.Login
```

Connecting a Module to a Database

You connect modules to the Pipeline Manager database and the BRM database through the Database Connect module. To do so:

1. Configure the Database Connect module in the **ifw.DataPool** section of the registry file.

   You can configure three types of connections:

   - A connection to the Pipeline Manager database.
   - A connection to the BRM database.
   - A connection to the database login queue (used by the DAT_Listener module).

2. When configuring a module that needs a connection to the Pipeline Manager database, use one of the following registry entries:

   - **DataConnection**
   - **IntegrateConnection**

   These entries do the same thing; they point to the **ifw.DataPool.Login** section. For example:

   ```
   DataConnection = ifw.DataPool.Login
   IntegrateConnection = ifw.DataPool.Login
   ```

   See the documentation for each module to determine which entry to use.

   > **Note:**
   >
   > Some modules can get data either from the database or from a file. If you configure the module to get data from a file, the module does not connect to the database.

3. When configuring a module that needs a connection to the BRM database, configure one of the following registry entries:

   - **DataConnection**
   - **InfranetConnection**

   These entries do the same thing; they point to the **ifw.DataPool.LoginInfranet** section. For example:

   ```
   DataConnection = ifw.DataPool.LoginInfranet
   InfranetConnection  = ifw.DataPool.LoginInfranet
   ```

Forcing a Database Reconnection

You can force the Database Connect module to reconnect to the Pipeline Manager database by using the following semaphore entry:

```
ifw.DataPool.Login.Module.Reconnect {}
```

This semaphore closes all open database connections and reconnects the Database Connect module to the Pipeline Manager database.

For information on how to create semaphore files, see "Updating Configuration Settings during Runtime by Using Semaphore Files".

Reloading Data into a Pipeline Manager Module

When you update data in the Pipeline Manager database, it is not automatically loaded into the modules. For example, if you change pricing data, EDRs continue to be rated by using the old pricing data until the new data is loaded into the data modules.

You use the **Reload** semaphore entry to reload data from the database into a module.

If the reload operation does not succeed, the module stops processing EDRs until data is loaded correctly. In some cases, you can configure how a module behaves if reloading fails:

- To configure a module to immediately resume processing using the previous data, set its **ReuseOnFailure** startup registry entry to **True**. Not all modules have this registry entry. Check the module's reference documentation to determine whether its registry includes **ReuseOnFailure**.
- To ensure that a module does not resume processing EDRs until the latest data is loaded, do not include **ReuseOnFailure** in the registry. This is the only option for modules that do not include this registry entry.

Using Business Parameter Settings from the BRM Database

You enable or disable optional BRM features and functionality by configuring business parameter settings, which are stored in **/config/business_params** objects in the BRM database. Pipeline Manager can determine whether these features and functionality are enabled by using the DAT_PortalConfig module, which retrieves and stores business parameter settings from the BRM database at pipeline initialization. Any other data modules that need a business parameter setting retrieve it directly from the DAT_PortalConfig module's internal memory.

Table 68-4 lists the data modules that use business parameter settings, the features that depend on the setting, and the **/config/business_params** parameter class and entry that each feature uses:

Table 68-4 Data Modules Using Business Parameter Settings

Pipeline Manager module	Feature	Parameter class	/config/business_params entry
DAT_AccountBatch	Balance monitoring.	**multi_bal**	**BalanceMonitoring**
DAT_BalanceBatch	Validity end time for first-usage balance elements.	**multi_bal**	**RestrictResourceValidityToOffer** **SortValidityBy** **CreditThresholdChecking**
DAT_Discount	Discount validity and exclusion rules.	**billing**	**ValidateDiscountDependency**
DAT_PriceModel	Pricing model for pipeline rating.	**subscription**	**TailormadeProductsSearch**
DAT_RatePlan	Charge for pipeline rating.	**subscription**	**TailormadeProductsSearch**

To set up Pipeline Manager to use business parameter settings from the BRM database, perform these tasks:

1. Configure the DAT_PortalConfig module in your registry file. This module must be listed in the registry file before any other data modules that are connected to it.

2. Configure data modules to retrieve business parameter settings from DAT_PortalConfig. See "Connecting Pipeline Manager Modules to DAT_PortalConfig".

After Pipeline Manager starts, you can:

- Verify that the entries loaded properly by printing the parameters that DAT_PortalConfig has stored in memory. See "Printing Business Parameter Settings Stored in DAT_PortalConfig Memory".
- Refresh business parameter settings stored in the DAT_PortalConfig module's internal memory. See "Refreshing Business Parameter Settings Stored in DAT_PortalConfig Memory".

Connecting Pipeline Manager Modules to DAT_PortalConfig

You must connect all data modules in your system that need business parameter settings to DAT_PortalConfig. You connect a module to DAT_PortalConfig by using the module's **PortalConfigDataModule** registry entry. For example:

```
PortalConfigDataModule=ifw.DataPool.PortalConfigDataModule
```

> **Note:**
>
> You can use any name you want to identify the registry section that configures DAT_PortalConfig, but you must use that name exactly when configuring modules to point to that registry section.

For example, the following entry, shown in bold, connects the DAT_Discount module to DAT_PortalConfig:

```
#------------------------------------------------------------
# Discount Data Module
#------------------------------------------------------------
DiscountModelDataModule
{
  ModuleName = DAT_Discount
  Module
  {
    InfranetConnection     = ifw.DataPool.LoginInfranet
    IntegrateConnection    = ifw.DataPool.Login
    PortalConfigDataModule = ifw.DataPool.PortalConfigDataModule
    AccountDataModule      = ifw.DataPool.CustomerData
  }
}
```

Printing Business Parameter Settings Stored in DAT_PortalConfig Memory

To print to a file the business parameter settings stored in the DAT_PortalConfig module's memory, use the CBPPrintData semaphore. For example:

```
ifw.DataPool.PortalConfig.Module.CBPPrintData=[Path][Filename]
```

where:

- *Path* specifies where to create the output file. By default, the file is created in the current directory.

- *Filename* specifies the name for the output file. The default file name is **DefaultCBPDataFile**_*timestamp*.**lst**. The module appends a timestamp to the end of the file name to prevent the module from overwriting existing files.

For example:

```
ifw.DataPool.PortalConfig.Module.CBPPrintData=Portal/text/prntdata
```

When you submit the print semaphore, DAT_PortalConfig generates an output file that uses the format shown below:

```
<BusParamConfiguration>
  <BusParamConfigurationList>
    <ParamClass name="group_name">
      <Param>
        <Name>parameter_name</Name>
        <Type>data_type</Type>
        <Value>parameter_value</Value>
      </Param>
    </ParamClass>
  </BusParamConfigurationList>
</BusParamConfiguration>
```

For example, the following shows a sample output file for the **billing** parameter class:

```
<BusParamConfiguration>
  <BusParamConfigurationList>
    <ParamClass name="billing">
      <Param>
        <Name>rerate_during_billing</Name>
        <Type>INT</Type>
        <Value>0</Value>
      </Param>
      <Param>
        <Name>validate_discount_dependency</Name>
        <Type>INT</Type>
        <Value>0</Value>
      </Param>
      <Param>
        <Name>sub_bal_validity</Name>
        <Type>INT</Type>
        <Value>0</Value>
      </Param>
    </ParamClass>
  </BusParamConfigurationList>
</BusParamConfiguration>
```

For information about semaphores, see "Using Semaphore Files to Control Pipeline Manager".

Refreshing Business Parameter Settings Stored in DAT_PortalConfig Memory

You must refresh DAT_PortalConfig memory whenever you update the **BalanceMonitoring**, **RestrictResourceValidityToOffer**, or **ValidateDiscountDependency** business parameter settings in the BRM database.

You refresh the memory by using the CBPReload semaphore entry. For example:

```
ifw.DataPool.PortalConfigDataModule.Module.CBPReload{}
```

For information about semaphores, see "Using Semaphore Files to Control Pipeline Manager".

Connecting a Pipeline Manager Module to Another Module

Most function modules connect to data modules to get configuration data. For example, the FCT_Account module requires a connection to the DAT_AccountBatch module. Also, some data modules connect to other data modules.

To connect one module to another, you configure a registry entry for the module that requires the connection. For example, to connect the FCT_Account module to the DAT_AccountBatch module, you enter this when you configure the FCT_Account module:

```
DataModule = ifw.DataPool.CustomerData
```

CustomerData identifies the DAT_AccountBatch module, which is configured in the registry like this:

```
#------------------------------------------------------------
# Infranet Customer Data
#------------------------------------------------------------
CustomerData
{
   ModuleName = DAT_AccountBatch
```

> **Note:**
>
> You can use any name you want to identify the registry section that configures a module, but you must use that name exactly when configuring modules to point to that registry section.

A function module can connect to more than one data module. For example, the FCT_ApplyBalance module includes two data module connection entries:

```
DiscountDataModule = ifw.DataPool.DiscountModelDataModule
BalanceDataModule = ifw.DataPool.BalanceDataModule
```

In addition, function modules, like data modules, can require a connection to the Pipeline Manager or BRM database, for example:

```
DataConnection = ifw.DataPool.LoginInfranet
```

Configuring Pipeline Buffers

Pipeline Manager uses buffers to control the flow of data moving from one thread to another. For example, you insert a buffer block into the LOG module to temporarily store log data received from your thread before it is written by the logging thread to a file.

To insert a buffer, you configure the pipeline's or module's **Buffer**, **InputBuffer**, or **OutputBuffer** registry section. In each section, you specify the buffer's type and size. Pipeline Manager supports the following buffer types:

- Rogue Wave buffers. See "Using Rogue Wave Buffers".
- Block transfer buffers. See "Using Block Transfer Buffers on Solaris Systems".
- Array buffers. See "Using Array Buffers on Solaris Systems".

> **Important:**
>
> When configuring buffers in multiple function pools, each buffer must have a unique name.

Using Rogue Wave Buffers

By default, all buffers in Pipeline Manager are Rogue Wave buffers. These buffers are simple FIFO buffers of a configurable size. When a thread writes to or reads from a Rogue Wave buffer, it locks the entire buffer to ensure the integrity of the data. For example, if a Rogue Wave buffer has 15 containers, all 15 containers are locked when a thread accesses the buffer. Other threads must wait for the buffer to be unlocked before they can read or write data. For this reason, Rogue Wave buffers work best when only one thread will access the buffer.

> **Note:**
>
> If multiple threads will access the buffer, use a block transfer. See "Using Block Transfer Buffers on Solaris Systems".

When a thread attempts to write to a full buffer or read from an empty buffer, the thread sleeps before attempting to access the buffer again.

To use a Rogue Wave buffer, you specify only the size of the buffer, by using the **Size** registry entry. This entry, listed in Table 68-5, goes in the **Buffer**, **InputBuffer**, or **OutputBuffer** registry section.

Table 68-5 Rogue Wave Buffers Registry Entry

Registry entry	Description	Mandatory
Size	Specifies the size of the internal data buffer.	Yes

The following shows sample registry entries for a Rogue Wave buffer:

```
Buffer
{
    Size = 100
}
```

This registry example creates a Rogue Wave buffer with 100 containers.

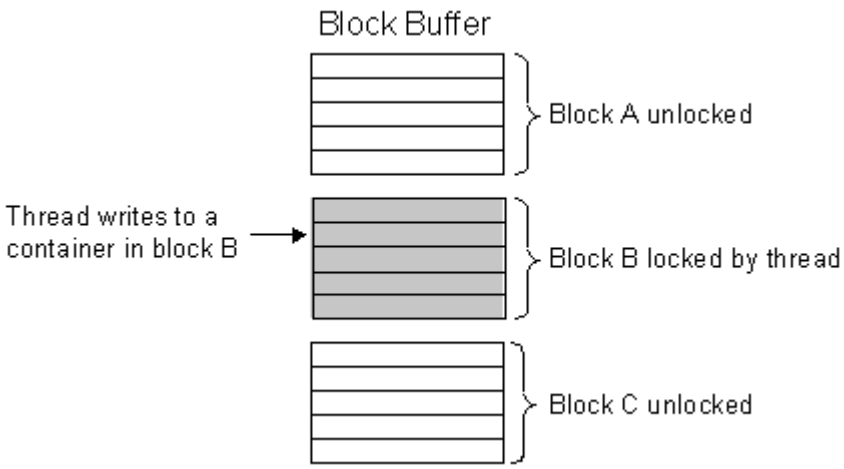
Using Block Transfer Buffers on Solaris Systems

Block transfer buffers address performance and scalability issues that occur when two or more threads are accessing the same buffer. They are recommended for use on Solaris systems.

A block transfer buffer is a buffer that is separated into logical subsections (or blocks) with a configurable number of buffer containers. When a thread accesses a buffer container, it locks only those containers that are in the same block. This enables other threads to access other buffer containers in the remaining free blocks.

For example, a buffer has 15 containers separated into 3 logical blocks. A thread writing to a container in block B locks the block to prevent other threads from changing the container's value during the write operation. Threads dedicated to blocks A and C can still read and write data because those blocks are unlocked as shown in Figure 68-1.

Figure 68-1 Block Transfer Buffer Locking



When a thread attempts to write to a full buffer or read from an empty buffer, the thread sleeps before attempting to access the buffer again.

To use a block transfer buffer, you use the buffer registry entries listed in Table 68-6. These entries go in the **Buffer**, **InputBuffer**, or **OutputBuffer** section.

Table 68-6 Solaris Block Transfer Buffers Registry Entries

Registry entry	Description	Mandatory
Size	Specifies the size of the internal data buffer.	Yes
BlockTransfer	Specifies whether the buffer operates in block transfer mode. **True**: The buffer operates in block transfer mode. **False**: The buffer does not operate in block transfer mode. If set to **False**, the pipeline ignores the **BlockSize** registry entry. The default is **False**.	Yes
BlockSize	Specifies the size of each buffer block.	Yes, if **BlockSize** is set to **True**.

The following shows sample registry entries for a block transfer buffer:

```
Buffer
{
  Size = 4000
  BlockTransfer = True
  BlockSize = 500
}
```

This example specifies a buffer size of 4,000 containers and a block size of 500 containers. Therefore, the buffer has eight (4,000/500) blocks.

Using Array Buffers on Solaris Systems

Array buffers address performance and scalability issues that occur when two or more threads are accessing the same buffer. They are recommended for use on Solaris systems.

Array buffers are similar to Rogue Wave buffers, except threads never lock the buffer. To protect shared data, threads use a compare and swap (CAS) method that atomically compares a container's old and current values before writing new data. This enables a thread to read data from a buffer container, modify it, and write it back only if no other thread modified it in the meantime.

When writing data to an array buffer, a thread performs the following:

1. Reads the buffer container. The thread takes three arguments: the container's address, the container's current value, and the new value.
2. Updates the local variable with the new value.
3. Determines whether any other thread modified the container in the interim by comparing the container's current value with the value it initially read in step 1:
   - If the value *has not* changed, the thread writes the new value to the container.
   - If the value *has* changed, another thread modified the container during the write operation. The thread does not change the value and instead starts the process over at step 1.

When a thread attempts to write to a full buffer or read from an empty buffer, the thread spins for a specified number of times and then enables another thread to access the buffer before spinning again. You can specify the maximum number of times the thread yields before sleeping. The thread sleeps for a specified amount of time before starting the spin-and-yield process again.

For example, if the maximum number of spins is 2, the maximum number of yields is 2, and the sleep time is 10 milliseconds, the thread performs the following while waiting for a buffer container to become available:

1. Spins 2 times.
2. Yields to another thread.
3. Spins 2 times.
4. Yields to another thread.
5. Sleeps for 10 milliseconds.
6. Starts over at step 1.

To use an array buffer, you use the buffer registry entries listed in Table 68-7. These entries go in the **Buffer**, **InputBuffer**, or **OutputBuffer** section.

Table 68-7 Solaris Array Buffers Registry Entries

Registry entry	Description	Mandatory
Size	Specifies the size of the internal data buffer.	Yes

Table 68-7 (Cont.) Solaris Array Buffers Registry Entries

Registry entry	Description	Mandatory
ArrayType	Specifies whether the buffer is an array buffer. **True**: The buffer is an array buffer. **False**: The buffer is not an array buffer. If set to **False**, the pipeline ignores the **SpinCount**, **YieldCount**, and **SleepTimeMilliSec** registry entries. The default is **False**.	Yes
SpinCount	Specifies the maximum number of times the thread spins while waiting for a buffer container to become available.	No
YieldCount	Specifies the maximum number of times a thread yields to another thread before the thread starts a sleep cycle.	No
SleepTimeMilliSec	Specifies how long the thread sleeps, in milliseconds, before trying to access the buffer again.	No

The following shows sample registry entries for an array buffer:

```
Buffer
{
  Size = 100
  ArrayType = True
  SpinCount = 100
  YieldCount = 100
  SleepTimeMilliSec = 10
}
```

Using Semaphore Files to Control Pipeline Manager

You use semaphore files to configure and control Pipeline Manager during runtime. They enable you to perform business tasks regularly without having to stop and restart the pipeline. For example, you can use semaphore files to stop a module or to reload data from the database.

The Controller checks for new semaphore files to process at a regular interval. You configure where and how often the Controller checks for new semaphore files by using the **Semaphore** and ProcessLoopTimeout registry entries.

When the Controller finds a semaphore file, it:

1. Prevents new transactions from being created.
2. Finishes processing all open transactions in the framework.
3. Stops the pipeline framework.
4. Loads the semaphore file into memory.
5. Changes the specified configuration settings and/or runs the specified semaphores.
6. Logs any processing errors in the **process.log** file.
7. Renames or deletes the semaphore file from the directory.

   You configure the Controller to rename or delete semaphore files by using the RetainFiles semaphore entry.
8. Stops and restarts the pipeline framework.

For information on creating semaphore files, see "Updating Configuration Settings during Runtime by Using Semaphore Files".

Updating Configuration Settings during Runtime by Using Semaphore Files

To change the Pipeline Manager configuration during runtime, you must:

1. Specify where and how often the Controller checks for semaphore files. See "Configuring Where and How Often the Controller Checks for Semaphore Files".

> **Note:**
>
> You perform this procedure only once, when you first configure your registry file.

2. Create your semaphore files. See "Procedure for Updating Configuration Settings".

Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to for system administration tasks. See "Using Perl Scripts to Administer Pipeline Manager".

Configuring Where and How Often the Controller Checks for Semaphore Files

You use the following registry entries in Table 68-8 to specify where and how often the Controller checks for semaphore files:

Table 68-8 Controller Configuration Registry Entries

Semaphore	Value	Description	Mandatory
ifw.ProcessLoopTimeout	Integer	Specifies the interval, in seconds, between polling for a new semaphore file. **Note:** This parameter controls the overall event loop, which includes looking for semaphore files.	Yes
ifw.Semaphore.FilePath	String	Specifies the directory where the Controller checks for semaphore files.	Yes
ifw.Semaphore.FileName	String	Specifies the name of the semaphore file.	Yes
ifw.Semaphore.RetainFiles	**True** **False**	Specifies whether semaphore files are deleted or saved after they are processed. • **True** specifies to save semaphore files. The Controller renames the file by appending the current timestamp to the file name in the format *YYYYMMDD_hhmmss* and logs the semaphore file's new name in the **process.log** file. For example, the **semaphore.reg** file is renamed **semaphore.reg_20031022_120803**. • **False** specifies to delete semaphore files immediately after they are processed. The default is **False**.	No

Sample Registry Entries

```
ifw
{
    ...
    ProcessLoopTimeout = 30
```

```
    ...
    Semaphore
    {
        FilePath = /BRM_home/ifw/semaphores
        FileName = semaphore.reg
        RetainFiles = True
    ...
    {
}
```

Procedure for Updating Configuration Settings

To update configuration settings during runtime:

1. Create a semaphore file using the file name specified in the registry file. (The examples in this chapter use **semaphore.reg**.)
2. Add new configuration or semaphore entries to the file. See "Semaphore File Syntax".

> **Note:**
>
> The maximum number of entries you can add is **10000**.

3. Copy the semaphore file to the semaphore directory.

> **Note:**
>
> - Some settings in the registry file cannot be configured by using semaphore files. For a list of commands that can be submitted by using semaphores for a particular module, see the *Semaphore file entries* section in the documentation for the module.
> - Before you submit a semaphore to Pipeline Manager, be sure that Pipeline Manager has finished starting up. (It displays the message **Ready for processing**.) If a semaphore file is submitted when Pipeline Manager is still starting, the system renames the semaphore file, logs a message that the semaphore file was renamed, and ignores the renamed file. The file is left in the semaphore input directory. To run the semaphore after the system completes startup, rename the file manually.
> - If a pipeline fails to process an update semaphore, the pipeline stops. To start it again, you must send another semaphore.

Semaphore File Syntax

Semaphore commands use one of these formats:

- Key-value pair format, such as **LoadZoneDescription = True**. These semaphore commands require a value.

> **Note:**
>
> The semaphore command fails if you do not supply a value.

- Semaphore entry { } format, such as **Reload{}**.

The commands in the semaphore file can be expressed in a nested hierarchy format or in a flattened syntax that uses periods to delimit nested sections. The syntax of a command reflects the hierarchical structure of the registry.

> **Note:**
>
> You must specify the full path for the command when using either the hierarchy or the flattened format.

The following examples show how to set the process log file name by using the hierarchy and flattened formats.

Hierarchy Format

```
ifw
{
    ProcessLog
    {
        Module
        {
            ITO
            {
                 FileName = process
            }
        }
    }
}
```

Flattened Format

```
ifw.ProcessLog.Module.ITO.FileName = process
```

Though registry files can vary in structure, commands for each type of module follow a similar pattern. For function modules, the syntax follows this pattern (shown in flattened format):

```
ifw.Pipelines.Pipeline_Name.Functions.Function_pool_name.
FunctionPool.Module_identifier.Module.Entry = Value
```

For example:

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
Aggregate.Module.Active = False
```

For data modules, the syntax is:

```
ifw.DataPool.Module_identifier.Module.Entry = Value
```

For example:

```
ifw.DataPool.ZoneDataModule.Module.ZoneModels.
ZM_MOBILE = /data9/INTEGRATE/test/config/ZM_MOBILE-new.dat
```

You can specify multiple commands in one semaphore file by placing each command on a separate line. For example:

```
ifw.Pipelines.ALL_RATE.Active = True
ifw.ProcessLog.Module.ITO.FileName = process
```

> **Note:**
>
> Avoid using multi-command semaphore files unless you are sure that each command works without error when submitted in a single-command semaphore file. For more information, see "Semaphore Error Messages".

Semaphore Error Messages

When a semaphore command is run correctly, the registry entry is removed and a success message is written to the process log.

If no command in a semaphore file can be executed correctly, the warning message **Semaphore was not processed; check spelling** is written to the process log.

> **Note:**
>
> When processing a multi-command semaphore file, if at least one command in the file runs successfully, the pipeline does not log a message indicating that a command has failed.

For more information on the process log, see "About Pipeline Manager Log Files".

Using Events to Start External Programs

To use pipeline events to trigger external programs, use the Event Handler.

Note:

- See the module reference documentation to find the events that a module sends.

  Events are named like this:

  - EVT_RELOAD_SUCCESSFUL
  - EVT_RELOAD_FAILED
- You can configure modules to send custom events to the Event Handler by using iScripts.

About Mapping Events to Programs

You map events to programs by using the registry file's **Event** subsection.

The **Events** subsection specifies the module and event combinations can trigger an external program. Use the following syntax to create the Events subsection:

```
Events
{
    ModuleSendingEvent
```

```
    {
        EventName = Action
        EventName = Action
        TimeToWait = WaitValue
    }
}
```

where:

- *ModuleSendingEvent* specifies the registry name of the module that sends the event to the Event Handler. Add an entry for each module that can trigger an external program.

  You can use wild cards (*) to specify multiple modules. For example, use **ifw.Pipelines.*** to specify all modules nested under the **ifw.Pipelines** section of the registry file.
- *EventName* specifies the event that triggers an external program. Add an entry for each event that triggers an external program.
- *Action* specifies the external program that is triggered by the event. Specify both the path and file name of the script or program.
- *WaitValue* specifies the time in seconds that the Event Handler waits for the external program to terminate. See "Controlling External Programs".

For example:

```
Events
{
    ifw.DataPool.Customer.Module
    {
        EVT_ReloadSuccess = ./script/script_1
        EVT_ReloadFailed = ./script/script_2
        TimeToWait = 30
    }
}
```

> **Note:**
>
> You cannot change this event-to-program mapping while Pipeline Manager is running. To map an event to a new script or change the existing mapping, you must edit the registry file and stop and restart Pipeline Manager.

Controlling External Programs

Use the **TimeToWait** registry entry to specify the time in seconds that the Event Handler waits for the external program to terminate. If the program does not terminate before the **TimeToWait** period ends, the external program is killed.

If an event is received while an external program is running, the event is queued and is started after the running program terminates.

When this option is specified, only one external program can be run at a time.

If **TimeToWait** is not enabled, the EventHandler does not wait for the external program to finish its job. Instead it starts new external programs depending on the events in the queue.

By default, no **TimeToWait** value is assumed.

About Running External Programs

The Event Handler can run only one external program at a time. If the Event Handler receives an event while an external program is running, it queues the event until the program terminates.

Troubleshooting Event Handling

You can log the events that a data module receives. This enables you to test event logging. To do so, set the data module's **LogEvents** registry entry to **True**. By default, event logging is off.

> **Note:**
>
> Not all data module support event logging. See the documentation for the data module that you are configuring.

About Pipeline Manager Transactions

Pipeline Manager uses transactional processing to ensure data integrity. When a system crash or power outage occurs, Pipeline Manager performs an automatic rollback and continues processing. Usually, the last CDR file that was being processed is rolled back and processed again.

In some cases, Pipeline Manager recognizes an inconsistent state of the file system; for example, an output file is missing. In these cases, Pipeline Manager does not restart and gives an error message.

> **Note:**
>
> A transaction can consist of one CDR file or multiple CDR files. You define the number of CDR files in a transaction by configuring the **UnitsPerTransaction** entry.

Pipeline Manager uses two components for transaction handling:

- The Transaction Manager handles transactions for a single pipeline. See "About the Transaction Manager".
- The Transaction ID Controller manages transaction IDs for the entire Pipeline Manager instance. See "Configuring the Transaction ID Controller".

About the Transaction Manager

The Transaction Manager is a mandatory pipeline component that coordinates the state of all transactions in one pipeline.

The Transaction Manager performs the following functions:

- Monitors a transaction's state. Transactions move through these three states:
  - Opened (started)

  - Prepared
  - Closed (ended)
- Persists state information to the binary log file. For information, see "About Transaction Log Files".

When a transaction is in progress, the following occurs:

1. The Input Controller notifies the Transaction Manager that a transaction started.
2. The Transaction Manager requests a transaction ID number from the Transaction ID Controller. See "Configuring the Transaction ID Controller".
3. The Transaction ID Controller issues the next ID number to the Transaction Manager.
4. The Input Controller, function modules, and Output Controller process the input stream and notify the Transaction Manager if any of the following are required:
   - Rollback. If a rollback is required, the Transaction Manager rolls back the transaction and undoes all changes.

     > **Note:**
     >
     > When redo is enabled, the Transaction Manager also cancels any newly opened transactions.

   - Cancel. If a cancel is required, the Transaction Manager undoes all changes made during the transaction.
5. The Output Controller notifies the Transaction Manager that the transaction ended.
6. The Transaction Manager requests the Input Controller, function modules, and Output Controller to prepare for a commit of the transaction.
7. The Transaction Manager performs one of the following:
   - If all of the modules prepare successfully, the Transaction Manager commits the transaction.
   - If the prepare fails, the Transaction Manager rolls back the transaction.

Two special types of EDRs are used for managing transactions:

- Before EDRs are processed, a *begin transaction EDR* is created. This tells Pipeline Manager which EDRs are part of the transaction.
- After all EDRs are processed, an *end transaction EDR* is created. When this EDR arrives at the output, the transaction can be committed.

You configure your Transaction Managers by using the **TransactionManager** section of the registry file.

About Cancelling Transactions When a Rollback Occurs

Use the Transaction Manager **RedoEnabled** registry entry to cancel all open transactions if a rollback occurs.

When a rollback is demanded, the Transaction Manager performs the following:

1. Disables the creation of new transactions.
2. Rolls back all attached modules.

3. Cancels any open transactions.
4. Re-enables the creation of new transactions.

When **RedoEnabled** is disabled, the Transaction Manager only rolls back the attached modules.

About Transaction Log Files

All dynamic data, for example, aggregation results, call assembling records, and duplicate check data, is always kept in main memory. In addition, to ensure transactional integrity, data in memory has to be made persistent. To do so, transactional modules write data to work files. Data in the work files is used to record the status of the transaction.

Each Transaction Manager generates its own binary log file, which stores information about a pipeline's currently open transactions. The Transaction Manager writes information to the file when a transaction starts or changes state and deletes the transaction from the file when it ends. Thus, the file's size changes constantly.

The binary log file stores the following for each open transaction:

- The transaction's starting timestamp.
- Transaction ID number.
- The list of CDR files that make up the transaction.
- Whether any of the following occurred:
  - Rollback
  - Cancel
  - Redo
  - Prepare

You should regularly back up binary log files. These files are needed when you stop and restart Pipeline Manager to resolve any open transactions at the time of failure.

> **Note:**
>
> When you stop and restart Pipeline Manager after an ungraceful shutdown, the Transaction Manager commits all prepared transactions and rolls back all other uncommitted transactions.

Configuring the Transaction ID Controller

You configure the Transaction ID Controller by using the **ifw.TransactionIDController** section of the registry file.

About Storing IDs in Cache

When the Transaction ID Controller must cache a block of IDs, it does the following:

1. Accesses the state file or table for the increment value and last issued ID number.
2. Caches the next block of transaction IDs.

For example, if the last ID is 200 and the increment value is 100, the Transaction ID Controller caches IDs 201 through 300.

3. Resets the last ID number in the state table or file.

   In the example above, the Transaction ID Controller sets the last ID to 300.

You configure the number of IDs stored in cache by using the Increment registry entry.

About the Transaction ID State File and Table

The state file or table stores the last issued transaction ID number and the configured increment value. You configure where the data is stored by using the Source registry entry.

When you configure the Transaction ID Controller to use a file, the data is stored in the file and directory you specify in the registry.

When you configure the Transaction ID Controller to use the database, the data is stored in the IFW_TAM table, which is automatically created in the Pipeline Manager database by the Pipeline Manager installer.

> **Note:**
>
> If you configure the Transaction ID Controller to store IDs in the database, only one Pipeline Manager instance at a time can access the Pipeline Manager database. This can reduce transaction processing performance.

You should back up the transaction ID state file or table regularly. This state information is needed to ensure that your system continues to create unique, system-wide IDs when you stop and restart Pipeline Manager.

Configuring Sequence Checking

Sequence checking ensures that a CDR file is not processed more than once. You configure your Sequencers by using the **ifw.SequencerPool** registry entries, and you assign Sequencers to pipelines by using the pipeline **Output** registry entries.

Sequence Numbers in the Header Record

The Header record in the EDR container includes two fields for sequence numbers:

- SEQUENCE_NUMBER. This is a unique reference that identifies each file. It indicates the file number of the specific file type, starting at 1 and incrementing by one for each new file of that type sent. Separate sequence numbering must be used for test and chargeable data. Having reached the maximum value (999999), the number restarts at 1.

  > **Note:**
  >
  > In the case of retransmission, this number is not incremented.

- ORIGIN_SEQUENCE_NUMBER. This is the original file sequence number as generated the first time. It is the same as SEQUENCE_NUMBER, but is never changed. It is used as a reference to the original file, if any processor has changed the file sequence number.

Deciding Whether to Use Sequencers

You should add Sequencers to your system when:

- You want to check for duplicate CDR files.
- Your CDR software does not automatically generate sequence numbers.
- Your pipelines split CDR files into multiple output files.

About Sequence Checking

When performing sequence checking, the Sequencer:

1. Receives the CDR file from the input module.
2. Checks for duplicates by comparing the sequence number in the stream's header with the sequence numbers in the state file or state table. See "Sequencer Log Files and Log Tables".
   - When the number is a duplicate, the Sequencer rejects the CDR file and rolls back the transaction.
   - When the number is not a duplicate, it passes the transaction directly to the Output Collection module.
3. Checks for gaps in sequence numbers by comparing the sequence number in the stream's header with the last sequence number in the state file or state table. If the sequence number is more than one digit greater than the previous number, a gap is identified. The Sequencer logs a message and stores the unused number in the state file or state table. See "Sequencer State Files and State Tables".

> **Note:**
>
> By default, the Sequencer:
>
> - Enables gaps in sequence numbers (caused by canceled or rolled back transactions). You can direct the Sequencer to reuse these number gaps by using the **Controller.ReuseGap registry** entry.
> - Does not start the gap in sequence numbers from 0. For example, if the first sequence number is 3, the Sequencer does not start the gap for the skipped sequence numbers from 0 (that is, gap of 1, 2). You can direct the Sequencer to add a gap for the skipped sequence numbers starting from 0 by using the **Controller.UseGapAtStartup** registry entry.

To configure the Sequencer to perform sequence checking, set the SequencerType registry entry to **Check**.

About Sequence Generation

When performing sequence generation, the Sequencer:

1. Receives the CDR file from the input module.

2. Assigns the next sequence number to the output file. To obtain this number, the Sequencer reads the last generated sequence number in the state file or state table and increments it by one.

   This process continues for each CDR file until the maximum value is reached. For information, see "About Maximum and Minimum Sequence Numbers".

> **Note:**
>
> If you configure the Sequencer to reuse gap numbers, it assigns unused gap numbers to the output file before assigning new sequence numbers.

To configure the Sequencer to perform sequence generation, set the SequencerType registry entry to **Generation**.

About Maximum and Minimum Sequence Numbers

The Sequencer generates numbers by starting at the configured minimum value and then incrementing by one until it reaches the configured maximum value. After the Sequencer uses the maximum value, you must manually reset the sequence number to the minimum value.

For example, if the minimum value is 1 and the maximum value is 10,000, the Sequencer assigns 1 to the first output file, 2 to the second output file, 3 to the third output file, and so on. When the sequencer assigns 10,000 to the ten-thousandth output file, you must manually reset the sequence number to 1 by changing the following fields in the IFW_SEQCHECK table:

- Set the **seq_orignumber** field to **0**.
- Set the **seq_gapnumbers** field to **-1**.

> **Note:**
>
> To prevent the Sequencer from incorrectly rejecting files as duplicates after you manually reset the sequence number to the minimum value, remove all the rows from the IFW_SEQLOG_IN table.

To configure the maximum and minimum values, do one of the following:

- **State files**. Edit the **MaxSequenceNumber** and **MinSequenceNumber** entries in the state file. The default minimum value is 0; the default maximum value is 99999.
- **State tables**. Use Pricing Center or PCC to set these values.

About Recycled EDRs

CDR input files sometimes contain nonvalid EDRs, which are rejected by the pipeline. When you recycle the input file through a pipeline to process any rejected EDRs, the file's original sequence number is no longer correct. The Sequencer automatically assigns new sequence numbers to recycled files to prevent them from being rejected as duplicates.

About Sequencer Files and Tables

Each Sequencer generates its own state and logging information, which can be stored in files or tables. You configure where state and logging information is stored by using the registry file.

> **Note:**
>
> When you store state and logging information in files, the Sequencer checks for duplicates by comparing the current sequence number against the last checked sequence number only. When you use tables, the Sequencer compares the number against all previously checked sequence numbers. For this reason, Oracle recommends using tables for charge offerion systems and using files only when testing your system in a development environment.

When you configure Sequencers to store logging information in files, all logging and state data is stored in the file and directory you specify in the registry file.

When you configure Sequencers to use tables, all logging and state data is stored in the database tables listed in Table 68-9, which are automatically created by the Pipeline Manager installer:

Table 68-9 Sequencer Logging and State Data Database Tables

Table name	Description
IFW_PIPELINE	Stores information about pipelines.
IFW_SEQCHECK	Stores the state of the Sequencer.
IFW_SEQLOG_OUT	Stores sequence generation log information.
IFW_SEQLOG_IN	Stores sequence checking log information.

You use Pricing Center or PCC to provide input to IFW_SEQCHECK and to view log information stored in IFW_SEQLOG_OUT and IFW_SEQLOG_IN.

Sequencer State Files and State Tables

Sequencer state files and state tables store the following information:

- The last generated sequence number
- The last checked sequence number
- Maximum and minimum sequence numbers

You should back up state files and state tables periodically. This information is needed to ensure that your system does not process duplicate CDR files when you stop and restart Pipeline Manager.

Sequencer Log Files and Log Tables

Sequencer log files and log tables store an entry for each sequence number that is checked or generated.

> **Note:**
>
> When the Sequencer reaches the maximum generated sequence number, delete all log entries. Otherwise, your log will contain duplicates. For more information, see "About Maximum and Minimum Sequence Numbers".

> **Note:**
>
> Log files and log tables grow indefinitely, so you should trim them periodically to reduce disk usage.

Checking and Generating Sequence Numbers

You can use Sequencers to configure pipelines to check for duplicate CDR input files and to check for gaps in sequence numbers. You can also configure pipelines to use Sequencers to generate sequence numbers. For information, see "Configuring Sequence Checking".

To enable sequence checking or sequence generation in a pipeline, perform the following tasks:

1. Configure your Sequencers by editing the **SequencerPool** section of the registry file. Ensure you specify the following:
   - The Sequencer name.
   - Whether Sequencer data is stored in a database table or files.
   - How to connect to the database or the path and file name of the Sequencer files.
   - Whether the Sequencer performs sequence checking or sequence generation. Each Sequencer performs only one of these functions.
2. For sequence generation, set minimum and maximum sequence numbers by doing one of the following:
   - If you configured the Sequencer to store data in a *database*, use Pricing Center or PCC to set these values.
   - If you configured the Sequencer to store data in *files*, set the **MaxSequenceNumber** and **MinSequenceNumber** entries in the Sequencer state file. For information, see "About Maximum and Minimum Sequence Numbers".

     > **Note:**
     >
     > The default minimum value is 0, and the default maximum value is 99999.
3. Assign Sequencers to pipeline output streams:
   - To assign a sequence checker to an output stream, edit the Sequencer registry entry in the Pipeline Output Controller. Specify the name of the Sequencer assigned to the output stream:

     ```
     Output
     {
     ```

```
    ...
    Sequencer = SequenceCheckerName
    ...
}
```

- To assign a sequence generator to an output stream, edit the Sequencer registry entry in the output module. Specify the name of the Sequencer assigned to the output stream:

```
OutputStreamName
{
    ModuleName = OUT_GenericStream
    Module
    {
        Sequencer = SequenceGeneratorName
    }
}
```

Configuring the NET_EM Module for Real-Time Processing

You can use Pipeline Manager for real-time discounting, real-time zoning, and real-time rerating.

The NET_EM module provides a link between the Connection Manager (CM) and the pipelines. You configure the NET_EM module in the data pool.

To configure the NET_EM module, you configure connection information such as the port number and threads, and you configure the **OpcodeName** section for each type of real-time processing: discounting, rerating, and zoning.

In this example, you configure the real-time discounting by specifying the PCM_OP_RATE_DISCOUNT_EVENT opcode:

```
ifw
{
...
    DataPool
    {
        RealtimePipeline
        {
            ModuleName = NET_EM
            Module
            {
                ThreadPool
                {
                    Port = 14579
                    UnixSockFile = /tmp/rerating_em_port
                    Threads = 2
                }
                DiscountOpcode
                {
                    OpcodeName = PCM_OP_RATE_DISCOUNT_EVENT
                    NumberOfRTPipelines = 2
                    PipelineName = DiscountPipeline
                }
            }
        }
    }
}
```

Each NET_EM module can perform one type of processing; for example, discounting, rerating, or zoning. You must configure a separate instance of Pipeline Manager for each NET_EM module.

You can configure multiple instances of the same type of NET_EM processing, for example, multiple rerating Pipeline Manager instances. You can then configure the CM to point to all the NET_EM modules. When multiple rerating pipeline instances are configured, the NET_EM module routes rerate requests to whichever of these pipeline instances is available.

To configure the NET_EM module:

1. Configure the NET_EM module in the registry. See "Configuring the NET_EM Module".
2. Configure the CM to send data to the NET_EM module. See "Configuring the CM to Send Real-Time Requests to the NET_EM Module".

Configuring the NET_EM Module

The NET_EM module receives various types of requests from the CM and routes the requests to the appropriate pipeline.

Specifying the Type of NET_EM Opcode Processing

To specify the type of processing the NET_EM module is used for, use the **OpcodeName** entry.

For real-time discounting, use:

```
OpcodeName = PCM_OP_RATE_DISCOUNT_EVENT
```

For real-time zoning, use:

```
OpcodeName = PCM_OP_RATE_GET_ZONEMAP_INFO
```

For real-time rerating, use:

```
OpcodeName = PCM_OP_RATE_PIPELINE_EVENT
```

Configuring the CM to Send Real-Time Requests to the NET_EM Module

To configure the CM to send rerate requests to the NET_EM module:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
2. For real-time rerating, ensure the following entry is *uncommented*:

   - cm fm_module *BRM_home*/**lib/fm_rerate.so fm_rerate_config - pin**
3. Edit the discounting **em_group** entry:

   ```
   - cm em_group em_type Opcode_name
   ```

   where:

   - *em_type* is the type of real-time processing; for example, discounting, zoning, or rerating. You can enter any string up to 15 characters. This entry must match the entry in the **em_pointer** entry.
   - *Opcode_name* is the opcode used.

   For discounting, use:

   ```
   - cm em_group discounting PCM_OP_RATE_DISCOUNT_EVENT
   ```

For zoning, use:

```
- cm em_group zoning PCM_OP_RATE_GET_ZONEMAP_INFO
```

For rerating, use:

```
- cm em_group rating PCM_OP_RATE_PIPELINE_EVENT
```

4. Edit the discounting **em_pointer** entry to match your environment, for example:

```
- cm em_pointer discounting ip cm_host 11945
- cm em_pointer zoning ip cm_host 11945
- cm em_pointer rating ip cm_host 11945
```

Instructions for this entry are included in the file.

You can enter multiple **em_pointer** entries. If the first NET_EM module is unavailable, the CM connects to a different NET_EM module.

> **Note:**
>
> To run multiple NET_EM instances, you must run multiple instances of Pipeline Manager. You use only one NET_EM module for each instance of Pipeline Manager.

5. Save the file.
6. Stop and restart the CM.

About Pipeline Manager Log Files

The log module is an optional pipeline component that generates and manages your system log files, which consist of the logs listed in Table 68-10:

Table 68-10 Pipeline Manager Log Files

Log file	Description
Process log	Contains general system messages for the pipeline framework, such as startup, shutdown, version numbers of modules, and semaphore file messages. The module generates one process log for the entire pipeline framework.
Pipeline log	Contains messages for one pipeline, such as the opening and closing of batch files, the number of processed EDRs, and statistics. The module generates one pipeline log file per pipeline.
Stream log	Contains detailed messages for one output stream. The module generates one stream log file per input stream. It contains all single error messages for the stream and event; for example, *zone data not found*. **Note:** The number of stream log files grows indefinitely, so you should delete them periodically to save disk space.

You configure your system log files by editing the registry file. You create a set of log module registry entries for each type of log file you want your system to generate. For example, to configure your system to generate all three system log files, you create one set of entries for the process log, one set for the pipeline log, and one set for the stream log.

- You configure the process log in the **ProcessLog** registry section.

- You configure the pipeline log in the **PipelineLog** registry section for each pipeline.
- You configure the stream log in the **OutputLog** registry section for each pipeline.

In addition to the log files handled by the log module:

- All processed sequence numbers of the EDR streams are logged in the sequence log file. See "Sequencer Log Files and Log Tables".
- All processed transactions are logged in the transaction log file. See "About Transaction Log Files".

Pipeline Manager Log File Registry Entries

The registry entries listed in Table 68-11 control Pipeline Manager log files.

Table 68-11 Pipeline Manager Log File Registry Entries

Entry	Module	Log file	Description
BalanceLockStatusLog	DAT_BalanceBatch	Process log	Specifies that when an event transaction is locked by an EDR transaction, it is logged to the Process log.
BinaryLogFileName	Transaction Manager	User specified	Specifies the path and file name of the binary log file, which is used to persist and restore open transactions.
InfranetPool	DAT_ConnectionPool		Specifies whether to log debug messages.
LogEvents	DAT_AccountBatch DAT_BalanceBatch DAT_Listener DAT_PriceModel DAT_Rateplan DAT_Recycle DAT_ResubmitBatch	Pipeline log	Specifies whether received events should be written to a log file. Use this entry to troubleshoot Pipeline Manager event handling.
Logging	FCT_Opcode	Pipeline log	Logs each opcode called from the processing pipeline.
LogTestResults	FCT_Suspense		Determines whether the results of test recycling are logged.
LogTransactions	DAT_BalanceBatch	Process log	Specifies if the balances affected during the CDR processing are logged.

Table 68-11 (Cont.) Pipeline Manager Log File Registry Entries

Entry	Module	Log file	Description
LogZoneModelNotFoundEntries	FCT_USC_Map	Stream log	Specifies that all log entries in INF_NO_USC_MAPPING_ENTRY are logged into the Stream log.
RecycleLog	FCT_Recycle		Specifies the log file parameters.
WriteToLogEnabled	Transaction Manager	Pipeline log	Specifies whether the Transaction Manager writes status information to the pipeline log file.

About Error Message Files

You use error message files to define the errors generated by your pipeline modules. All modules have their own error message file (**.msg**), which is installed by default in the *pipeline_home***/etc** directory.

The default error message files already define all of the module error codes, but you can add custom error codes or change the existing definitions by editing the files.

Error message file entries use the following format:

```
[messageName] | [messageText] | [messageNumber]
```

where:

- *messageName* specifies the module error code. For example, ERR_WRITE_FILE.
- *messageText* specifies the message text to write to the log file.
- *messageNumber* specifies the error number to write to the log file. The default is **0**.

For example, the DAT_AccountBatch module uses the *pipeline_home***/etc/DAT_AccountBatch.msg** message file. This file includes the following entries:

```
ERR_LISTENER_NOT_FOUND    | Listener '%s' not found.|30013
INF_STARTED_LOADING       | Started loading account data.|30024
INF_ENTRIES_LOADED        | %s %s loaded.|30025
INF_FINISHED_LOADING      | Finished loading account data.|30026
```

> **Note:**
>
> The LOG module ignores comments, which start with a pound symbol (#).

About Log File Contents

The LOG module logs the following information to the system log file in ITO format:

- Date

- Time
- Node
- Application name
- Message group
- Severity
- Error number
- Text

> **Note:**
>
> All fields are separated by blanks.

For example:

```
03.10.2002 08:18:42 system ifw INTEGRATE NORMAL
00000 - No registry entry 'MultiThreaded(default is true)' found.
```

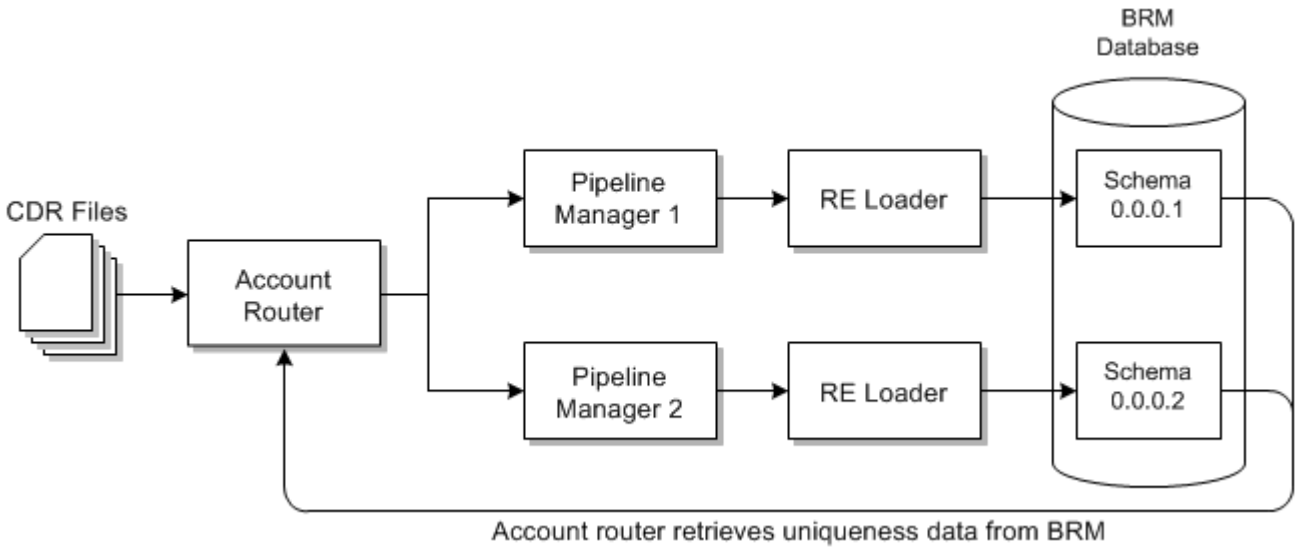
Using Pipeline Manager with Multiple Database Schemas

When you use Multidatabase Manager, you run an instance of Pipeline Manager for each BRM database schema. The following types of data must be managed:

- **Account data:** When account data changes in a database, the Account Synchronization Data Manager (DM) sends the data to Pipeline Manager. The DAT_Listener module map file specifies which Pipeline Manager instance the data is sent to.
- **CDR data:** Before a call details record (CDR) file is processed, the Pipeline Manager FCT_AccountRouter module separates CDRs by BRM database schemas and sends the CDRs to the appropriate Pipeline Manager instance.

Figure 68-2 shows how FCT_AccountRouter manages incoming CDRs:

Figure 68-2 FCT_AccountRouter CDR Management



Troubleshooting Pipeline Modules

You can troubleshoot problems in the pipeline modules by writing the contents of the EDRs generated by various pipeline modules into a log file. The file shows how each module accessed the EDR and the changes each module made to the EDR. You can read the log file to check if the pipeline modules processed the EDRs as expected and correct any problems you find.

Use the **EdrTrace** entry in the pipeline registry file to write the contents of the EDR to a file. You can configure **EdrTrace** to write the EDR contents to a file for specific modules that you want to debug. The **EdrTrace** entry includes the parameters listed in Table 68-12:

Table 68-12 EdrTrace Log File Registry Entries

Entry	Description
EDRTraceEnabled	Enables or disables EDR trace: • **True** enables EDR trace. • **False** disables EDR trace. The default is **False**.
EdrTrace	Specifies the EDR trace.
TraceLog	Specifies the following information about the EDR log file: • **FilePath**. The path to the log file. The default is **/ifw/log/edrLog**. • **FileName**. The name of the log file. The default is **edrdump**. • **FilePrefix**. The prefix to the log file name. The default is **log_**. • **FileSuffix**. The log file name extension. The default is **.log**.
TraceStartPoint	Specifies the pipeline module from which you want to start logging the EDR contents. This registry entry is mandatory. The default is **Input.module**.
TraceEndPoint	Specifies the pipeline module up to which you want to log the EDR contents. The default is **Output.module**. **Important:** If both the **TraceStartPoint** and **TraceEndPoint** registry entries are specified, the EDR log file contains changes from all the modules from **TraceStartPoint** to **TraceEndPoint**. If only **TraceStartPoint** is specified, the EDR log file contains changes from the module specified in that entry up to the Output module. To log EDR changes for only one module, **TraceStartPoint** and **TraceEndPoint** must specify the same module.

Writing EDR Contents to a Log File

To write the contents of the EDR to a log file and use it to debug pipeline modules, include the **EdrTrace** entry by using the following syntax:

```
...
Output
{
...
  EdrTraceEnabled = value
  EdrTrace
  {
   TraceLog
   {
    FilePath = file_path
```

```
    FileName = file_name
    FilePrefix = prefix
    FileSuffix = suffix
   }
   TraceStartPoint = Functions.Processing.FunctionPool.start_module_name
   TraceEndPoint = Functions.Processing.FunctionPool.end_module_name
  }
}
```

where:

- *start_module_name* is the user-defined name or label of the pipeline module from where the logging of the EDR contents starts.
- *end_module_name* is the user-defined name or label of the last pipeline module for the logging of the EDR contents.

Using a Semaphore to Write EDR Contents to a File for Debugging

You can change the EDR trace by sending a semaphore to the Output Controller module at run time without stopping the pipeline. You can perform the following changes to the **EdrTrace** entry through a semaphore:

- Enable or disable logging the EDR contents.
- Change **TraceStartPoint** and **TraceEndPoint** for logging the EDR contents.

To change the EDR content logging at run time, send a semaphore with the following syntax:

```
ifw.Pipelines.pipeline_name.Output.EdrTrace
{
  TraceStartPoint = new_start_value
  TraceEndPoint = new_end_value
}
```

Sample EDR Content Log File

The following sample output of **EdrTrace** shows EDR contents from Input to Output modules:

```
= = = = B E G I N   T R A N S A C T I O N = = = =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME :  : test2.edr : setString
ifw.Pipelines.ALL_RATE.Input : INTERNAL.TRANSACTION_ID : 0.0 : 4 : setDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageType.Module : INTERNAL.TRANSACTION_ID :
4 :  : getDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApolloDiscountModule.Module :
INTERNAL.TRANSACTION_ID : 4 :  : getDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApolloApplyBalanceModule.Module :
INTERNAL.TRANSACTION_ID : 4 :  : getDecim
al
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceOutputSplit.Module :
INTERNAL.TRANSACTION_ID : 4 :  : getDecimal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ObjectCacheTypeOutputSplit.Module :
INTERNAL.TRANSACTION_ID : 4 :  : getDec
imal
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Rejection.Module : INTERNAL.TRANSACTION_ID :
4 :  : getDecimal
ifw.Pipelines.ALL_RATE.Output : INTERNAL.TRANSACTION_ID : 4 :  : getDecimal
= = = = B E G I N   C O N T A I N E R = = = =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME :  : test2.edr : setString
ifw.Pipelines.ALL_RATE.Input : INTERNAL.SEQ_CHECK : 0 : 1 : setLong
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : INTERNAL.STREAM_NAME :
test2.edr :  : getString
```

```
ifw.Pipelines.ALL_RATE.Output : INTERNAL.STREAM_NAME : test2.edr :  : getString
ifw.Pipelines.ALL_RATE.Output : INTERNAL.STREAM_NAME : test2.edr :  : getString
ifw.Pipelines.ALL_RATE.Output : INTERNAL.SEQ_GENERATION : 0 :  : getLong
ifw.Pipelines.ALL_RATE.Output : INTERNAL.OFFSET_GENERATION : 0 :  : getLong
=  =  =  =  C O N T A I N E R   H E A D E R  =  =  =  =
ifw.Pipelines.ALL_RATE.Input : HEADER.TRANSFER_CUTOFF_TIMESTAMP : 20061204000445 :  : getDate
ifw.Pipelines.ALL_RATE.Input : HEADER.IAC_LIST :  :  : getString
ifw.Pipelines.ALL_RATE.Input : HEADER.CC_LIST :  :  : getString
ifw.Pipelines.ALL_RATE.Input : HEADER.IAC_LIST : 00 : 00 : setString
ifw.Pipelines.ALL_RATE.Input : HEADER.IAC_LIST : 00 :  : getString
ifw.Pipelines.ALL_RATE.Input : HEADER.CC_LIST : 49 : 49 : setString
ifw.Pipelines.ALL_RATE.Input : HEADER.CC_LIST : 49 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module :
HEADER.QUERYABLE_FIELDS_MAPPING :  :  : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : HEADER.CREATION_PROCESS :
PREPROCESS_PIPELINE :  : get
String
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module : HEADER.BATCH_ID :  :  :
getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Suspense.Module : HEADER.BATCH_ID :  :  :
getString
=  =  =  =  C O N T A I N E R   D E T A I L  =  =  =  =
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.DURATION : 300 :  : getDecimal
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_END_TIMESTAMP : 20061115102400 : 20061115102400 : setDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_END_TIMESTAMP : 20061115102400 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.NE_CHARGING_END_TIMESTAMP : 20061115102400 : 20061115102400 :
setDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.NE_CHARGING_END_TIMESTAMP : 20061115102400 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.RETAIL_CHARGED_AMOUNT_VALUE : 0.0 :  : getDecimal
ifw.Pipelines.ALL_RATE.Input : DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE : 0.0 :  : getDecimal
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.CHARGING_START_TIMESTAMP : 20061115101900 :  : getDate
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_TYPE_OF_NUMBER : 0 :  : getLong
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_MODIFICATION_INDICATOR : 00 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 : 0049100052 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_TYPE_OF_NUMBER : 0 :  : getLong
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_MODIFICATION_INDICATOR : 00 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 : 0049100056 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.C_NUMBER :  :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.RECORD_TYPE : 020 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.A_NUMBER : 0049100052 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_A_NUMBER_ZONE : 0049100052 : 0049100052 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_A_NUMBER_ZONE : 0049100052 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.B_NUMBER : 0049100056 :  : getString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_B_NUMBER_ZONE : 0049100056 : 0049100056 : setString
ifw.Pipelines.ALL_RATE.Input : DETAIL.INTERN_B_NUMBER_ZONE : 0049100056 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module :
DETAIL.INTERN_PROCESS_STATUS : 0 :  : getLong
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module :
DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME.0 :  : ALL_RATE
: setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module :
DETAIL.ASS_SUSPENSE_EXT.SOURCE_FILENAME.0 :  : test3.e
dr : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module :
```

```
DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS.0 :  :  : set
String
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.BASIC_SERVICE :
TEL :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module :
DETAIL.INTERN_USAGE_CLASS :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module :
DETAIL.ASS_GSMW_EXT.LOCATION_AREA_INDICATOR.0 :  :
: getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module :
DETAIL.QOS_REQUESTED :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.QOS_USED :  :  :
getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module : DETAIL.RECORD_TYPE :
020 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module :
DETAIL.INTERN_SERVICE_CODE : TEL : TEL : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module :
DETAIL.INTERN_SERVICE_CLASS : DEF : DEF : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.USAGE_CLASS :
NORM :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.USAGE_TYPE :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.WHOLESALE_IMPACT_CATEGORY :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.TARIFF_CLASS :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.TARIFF_SUB_CLASS :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.RECORD_TYPE :
020 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.CONNECT_TYPE :
17 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.CONNECT_SUB_TYPE : 01 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.INTERN_C_NUMBER_ZONE :  :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module : DETAIL.USAGE_CLASS :
NORM :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module :
DETAIL.INTERN_USAGE_CLASS :  : NORM : setString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module : DETAIL.DISCARDING :
0 :  : getLong
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module : DETAIL.RECORD_TYPE :
020 :  : getString
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module :
DETAIL.SOURCE_NETWORK :  :  : getString
=  =  =  =  E N D    C O N T A I N E R  =  =  =  =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME :  : test3.edr : setString
=  =  =  =  E N D    T R A N S A C T I O N  =  =  =  =
ifw.Pipelines.ALL_RATE.Input : INTERNAL.STREAM_NAME :  : test3.edr : setString
ifw.Pipelines.ALL_RATE.Input : INTERNAL.TRANSACTION_ID : 0.0 : 6 : setDecimal
```

Using Perl Scripts to Administer Pipeline Manager

Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to start and stop various types of pipelines and perform other system administration tasks.

Table 68-13 describes the files and scripts used for controlling pipelines:

Table 68-13 Pipeline Manager Administration Perl Scripts

Semaphore and Perl script file names	Description
dump_portal_act_data.reg **dump_portal_act_data.pl**	Outputs account data for all accounts currently in memory. By default, data is written to the **cust.data** file, located in the directory where you launch Pipeline Manager. Runs the DAT_Account module **PrintData** semaphore.
off_queue_buffer.reg **off_queue_buffer.pl**	Disables logging of the messages processed by the queue. Sets the DAT_Listener module **LogEvents** entry to **False**.
reload_portal_act_data.reg **reload_portal_act_data.pl**	Reloads accounts from the BRM database. Runs the DAT_Account module **Reload** semaphore.
reload_price.reg **reload_price.pl**	Reloads all the pricings and charges. Runs the DAT_Price module **Reload** semaphore.
reload_zone.reg **reload_zone.pl**	Reloads all the zones and zone model data. Runs the DAT_Zone module **Reload** semaphore.
set_call_ass_limit.reg **set_call_ass_limit.pl**	Sets a new flush limit for call assembly (by default, 30). Runs the FCT_CallAssembling module **FlushLimit** semaphore.
set_dup_check_limit.reg **set_dup_check_limit.pl**	Sets a new limit for duplicate checking. If you do not specify any parameter, it sets the **BufferLimit** entry to three days before the current date, and it sets the **StoreLimit** entry to seven days before the **BufferLimit** date. This script creates and runs the **set_dup_check_limit.reg** semaphore. To modify the default **BufferLimit** and **StoreLimit** values, run the script with these two parameters: **set_dup_check_limit.pl** *buffer_limit store_limit* For example: `set_dup_check_limit.pl 5 5` In this example, if today is November 28, then the buffer limit is set to November 23 and the store limit is set to November 18.

69

Starting and Stopping Pipeline Manager

This chapter describes how to start and stop Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager framework components and pipelines.

Customizing the pin_ctl Utility Environment Variables for Pipeline Manager

Some BRM components need environment variables set before starting. You can edit the **pin_ctl.conf** file to change the environment variables if yours are different from the default settings.

1. Open the **pin_ctl.conf** file in *BRM_home***/bin**.
2. To define environment variables for pipeline registry files, find the following lines:

```
# registry details for pipeline services
aaa env_platform:common env_variable:AAA_REGISTRY env_val:$IFW_HOME/conf/diameter_charge.reg
rtp env_platform:common env_variable:RTP_REGISTRY env_val:$IFW_HOME/conf/wirelessRealtime.reg
bre env_platform:common env_variable:BRE_REGISTRY env_val:$IFW_HOME/conf/wireless.reg
```

3. Add the following line for each pipeline component that uses a registry file:

   component **env_platform:common env_variable:***registry_variable* **env_val:$IFW_HOME/***registry_file*

   where:

   - *component* is the pipeline component name.
   - *registry_variable* is the environment variable to set before starting *component*. The syntax for pipeline registry environment variables is ***_REGISTRY**.
   - *registry_file* is the path and file name for the pipeline registry file.

   For example:

```
aaa env_platform:common env_variable:AAA_REGISTRY env_val:$IFW_HOME/conf/diameter_charge.reg
```

4. Save and close the file.

Starting Pipeline Manager Components by Using pin_ctl

To add a custom Pipeline Manager component to the components list, use the **:pipeline** parameter. For example:

```
4 bre_custom:pipeline
```

To start a BRM component by using the **pin_ctl** utility:

1. Edit the *BRM_home***/bin/pin_ctl.conf** configuration file.
2. Copy the *pipeline_home***/bin/ifw** binary file to *pipeline_home***/bin/***component*, where component is the pipeline component name.
3. Go to the *BRM_home***/bin** directory.

4. Run the **pin_ctl** utility with the **start** action:

   ```
   pin_ctl start component
   ```

   where *component* is the component you want to start.

Starting and Stopping Pipeline Manager Manually

You can stop and start Pipeline Manager by using the command line instead of by using the **pin_ctl** utility.

> **Note:**
>
> If you start Pipeline Manager manually, you cannot use the **pin_ctl** utility to control Pipeline Manager (for example, to stop it or to get the status).

Starting Pipeline Manager

You start an instance of Pipeline Manager by using the following command from the *pipeline_home* directory:

```
bin/ifw -r RegistryFile
```

where *RegistryFile* is the name of the registry file.

> **Note:**
>
> If Pipeline Manager cannot establish a connection with the Pipeline Manager database (most likely because the database is down), you receive an error message and the Pipeline Manager startup is canceled.

The general syntax for the **ifw** command and parameters is:

```
ifw  -r RegistryFile | -h | -v [-r RegistryFile]
```

where:

-r *RegistryFile*

Starts Pipeline Manager with the specified registry file.

-h

Displays the syntax and parameters.

-v [**-r** *RegistryFile*]

Displays the version of Pipeline Manager. If you use the **-r** parameter, it also displays the version and name of data and function modules. For example:

```
ifw -v -r conf/wireless.reg
Module ifw.DataPool.Listener.Module Name: DAT_Listener, Version: 10010
```

Pipeline Manager displays **Ready for processing** when startup procedures are completed.

Stopping Pipeline Manager

You stop Pipeline Manager by using the following semaphore entry:

```
ifw.Active = FALSE
```

Starting and Stopping Individual Pipelines

When you start Pipeline Manager, the Controller starts all pipelines. However, you can stop or restart individual pipelines by using semaphores.

> **Note:**
>
> When a pipeline cannot establish a connection with the Pipeline Manager database (most likely because the database is down), you receive an error message and the pipeline startup is canceled.

To start an individual pipeline, use the following semaphore entry:

```
ifw.Pipelines.PipelineName.Active = True
```

where *PipelineName* is the name of the pipeline.

To stop an individual pipeline, use the following semaphore entry:

```
ifw.Pipelines.PipelineName.Active = False
```

where *PipelineName* is the name of the pipeline.

If files are added to the input directory after a pipeline is stopped and before it is restarted, the files are processed in order based on their last modified timestamp.

> **Note:**
>
> Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to start and stop various types of pipelines and perform other system administration tasks. See "Using Perl Scripts to Start and Stop Pipeline Manager".

Restarting Pipeline Manager after an Abnormal Shutdown

Some modules track data in data files. If an abnormal shutdown occurs, you must delete the data files that were in progress when the shut-down occurred.

Using Perl Scripts to Start and Stop Pipeline Manager

Pipeline Manager includes a set of Perl scripts, and associated semaphore files, that you can use to start and stop various types of pipelines and perform other system administration tasks.

Table 69-1 describes the files and scripts used for starting and stopping pipelines:

Table 69-1 Scripts for Starting and Stopping Pipelines

Semaphore and Perl script file names	Description
start_all_pipeline.reg **start_all_pipeline.pl**	Starts all pipelines configured in the **start_all_pipeline.reg** semaphore file. By default, this list includes these pipelines: • ALL_RATE • PRE_RECYCLE • PRE_PROCESS • ALL_BCKOUT • ALL_RERATE If you add custom pipelines, add their names to the semaphore file according to the default examples. **Important:** • Do not run rating pipelines (ALL_RATE, PRE_RECYCLE, and PRE_PROCESS) at the same time that you run the rerating pipeline (ALL_RERATE). Edit the script to specify which pipeline to run. • Before running ALL_RERATE, ensure that the backout pipeline (ALL_BCKOUT) has processed all EDRs that were extracted by the Event Extraction tool.
start_all_rate.reg **start_all_rate.pl**	Starts the ALL_RATE pipeline.
start_DiscountPipeline.reg **start_DiscountPipeline.pl**	Starts the discount pipeline (DiscountPipeline).
start_main_stop_rerating.reg **start_main_stop_rerating.pl**	Starts these pipelines: • PRE_PROCESS • PRE_RECYCLE • ALL_RATE Stops these pipelines: • ALL_BCKOUT • ALL_RERATE
start_pre_process.reg **start_pre_process.pl**	Starts the preprocessing pipeline (PRE_PROCESS).
start_pre_recycle.reg **start_pre_recycle.pl**	Starts the pre-recycle pipeline (PRE_RECYCLE).
start_RealtimePipelineGPRS.reg **start_RealtimePipelineGPRS.pl**	Starts the real-time GPRS pipeline (RealtimePipelineGPRS).
start_RealtimePipelineGSM.reg **start_RealtimePipelineGSM.pl**	Starts the real-time GSM pipeline (RealtimePipelineGSM).
start_RealtimePipelineZone.reg **start_RealtimePipelineZone.pl**	Starts the real-time zoning pipeline (RealtimePipelineZone).
start_recycle.reg **start_recycle.pl**	Starts recycling rejected CDRs.
start_rerating_stop_main.reg **start_rerating_stop_main.pl**	Stops these pipelines: • ALL_RATE • PRE_RECYCLE • PRE_PROCESS Starts these pipelines: • ALL_BCKOUT • ALL_RERATE

Table 69-1 (Cont.) Scripts for Starting and Stopping Pipelines

Semaphore and Perl script file names	Description
stop_all_pipeline.reg **stop_all_pipeline.pl**	Stops all the pipelines configured in the **stop_all_pipeline.reg** semaphore file. By default, this list includes these pipelines: • ALL_RATE • PRE_RECYCLE • PRE_PROCESS • ALL_BCKOUT • ALL_RERATE If you add custom pipelines, add their names to the semaphore file according to the default examples.
stop_all_rate.reg **stop_all_rate.pl**	Stops the ALL_RATE pipeline.
stop_DiscountPipeline.reg **stop_DiscountPipeline.pl**	Stops the discount pipeline (DiscountPipeline).
stop_pre_process.reg **stop_pre_process.pl**	Stops the preprocessing pipeline (PRE_PROCESS).
stop_pre_recycle.reg **stop_pre_recycle.pl**	Stops the pre-recycle pipeline (PRE_RECYCLE).
stop_RealtimePipelineGPRS.reg **stop_RealtimePipelineGPRS.pl**	Stops the real-time pipeline for GPRS (RealtimePipelineGPRS).
stop_RealtimePipelineGSM.reg **stop_RealtimePipelineGSM.pl**	Stops the real-time pipeline for GSM (RealtimePipelineGSM).
stop_RealtimePipelineZone.reg **stop_RealtimePipelineZone.pl**	Stops the real-time pipeline for zoning (RealtimePipelineZone).

70

Monitoring Pipeline Manager

This chapter describes how to monitor Oracle Communications Billing and Revenue Management Pipeline Manager.

Monitoring Pipeline Manager

For information about improving Pipeline Manager performance, see "Optimizing Pipeline Manager Performance".

Monitoring Pipeline Manager Memory Usage

You can use the MemoryMonitor module to monitor Pipeline Manager memory during startup and while it is processing files. You set a threshold for the amount or percentage of memory that determines when Pipeline Manager should issue a warning or gracefully shut down. You can set the thresholds as a percentage or as kilobytes or megabytes.

For example, if you set **ShutdownFreeMemLimit** to 50 and **ScaleUnit** to **M**, Pipeline Manager shuts down gracefully when the remaining free system memory reaches 50 MB. If you set **WarningFreeMemLimit** to 10 and **ScaleUnit** to **P**, Pipeline Manager logs a warning when the remaining free system memory reaches 10 percent.

Monitoring Pipeline Manager EDR Throughput

You can monitor the following statistics for each pipeline:

- Number of event data records (EDRs) since startup.
- Accumulated EDR processing time since startup.
- Total number of EDRs since startup, independent of any transaction. This number is incremented after every processed EDR.
- Total number of EDRs after the transaction ended. This number is not incremented until the current transaction has ended.
- The real-time EDR count increments after each EDR is processed, while the transaction count increments EDR count only after transaction/file processing is ended.
- Number of transactions since startup.
- EDRs per second (throughput). This data includes the timestamp of when the measurement was taken.

You can use the Operations Management Framework (OMF) HTTP protocol to access the data.

Getting Recent Pipeline Log File Entries

You can display recent log file entries in the OMF HTTP server. The entries are also included in the Diagnostic Data Handler output file.

The log messages are stored in a circular buffer that stores the last 1000 log messages.

You can change the number of error messages stored in the buffer. To do so, edit the **CircularBufferSize** registry entry in the **ITO** section.

For example:

```
ProcessLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            LogLevel            = Debug
...
            CircularBufferSize  = 100
         }
```

Using the pin_db_alert Utility to Monitor Key Performance Indicators

KPIs are metrics you use to quantify the health of your database and to alert you when potential issues exist. They identify database tables that must be archived or purged and indexes, triggers, and stored procedures that are missing or invalid.

The PROCEDURES KPI is used for Pipeline Manager.

The **proceduresList** module retrieves a list of stored procedures in the BRM system and writes the stored procedure names and status (VALID or INVALID) to the **proceduresList_PROCEDURES.out** file. This enables Pipeline Manager to compile data in parallel and to restore it from the precompiled data files file.

The **proceduresList_validation** module compares the list of stored procedures in the **proceduresList** validation configuration file to the procedures in the results file and writes missing procedures to the **proceduresList_validation_PROCEDURES.out** file.

Important: If you installed optional managers that use unique stored procedures or if you created custom stored procedures, you must add them to the **proceduresList** validation configuration file to monitor their status.

KPIs are monitored when you run the **pin_db_alert.pl** utility.

Collecting Diagnostic Information by Using RDA

Remote Diagnostic Agent (RDA) is an Oracle standard tool used to collect diagnostic data from your system applications environment.

You can use RDA to collect BRM and Pipeline Manager diagnostic information. The information collected from Pipeline Manager includes:

- Configuration files

  RDA collects the pipeline configuration data from the **.reg** (registry) and **.dsc** (description) files. For example, RDA collects the configuration data for wireless from the **wireless.reg** and **containerDesc.dsc** files.

- Log files

RDA collects pipeline log data from the process log, pipeline log, and stream log files. For example, RDA collects the log data for wireless from the **processWIRELESS.log** file, the **log_streamRT1.log** file, and so on.

- Other files

  RDA collects pipeline installation and version details from the **vpd.properties** and **piperev.dat** files.

To find Pipeline Manager information, RDA looks at the registry files. To collect pipeline log files, verify that the INT_HOME environment variable is set to the Pipeline Manager installation directory.

71

Optimizing Pipeline Manager Performance

This chapter describes tools and techniques you can use to optimize Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager performance.

Pipeline Manager Optimization Overview

When you optimize Pipeline Manager performance, your objective is to increase the percentage of CPU time spent on user processes and to decrease the percentage of time spent idle or on system processes.

Complete performance tuning requires much testing. Due to the complexity of most Pipeline Manager configurations, optimization is a highly iterative process. You cannot configure options formulaically, but you must test many configurations and then implement the optimal configuration. This chapter describes optimization methods to guide your testing for a given set of hardware resources.

Software optimization techniques can include modifying the following:

- The number and type of function modules.
- The design of custom iScripts and iRules.
- The number of system threads used by a pipeline.
- The number of call data record (CDR) files configured for a transaction.
- The number of pipelines configured for the system.

> **Note:**
>
> Available hardware resources can constrain the usefulness of some optimization techniques. For example, if your system has only a few CPUs, you probably will not see performance gains by using multithreaded mode.

Key Metrics for Measuring Performance

When evaluating performance improvement, the primary metrics to monitor are:

- The ratio of CPU time spent on system processes to CPU time spent on user processes. This ratio should be about 1 to 2 or lower.
- The percentage of idle CPU time. This percentage should be 20 percent or less.
- The results of performance tests using sample CDR files.

About Measuring Pipeline Manager Performance

You use the Pipeline Manager instrumentation feature as the primary tool for measuring Pipeline Manager performance. See "Measuring System Latencies with Instrumentation" for more information. When instrumentation is enabled, information about how much time in

microseconds each function module uses to process a certain number of files is written to the pipeline log file (**pipeline.log**). You then use this information when you apply some optimization techniques.

Other Pipeline Manager performance monitoring tools are:

- Monitor event data record (EDR) throughput. See "Monitoring Pipeline Manager EDR Throughput".
- Monitor recent log files. See "Getting Recent Pipeline Log File Entries".
- Monitor memory usage.

Information Requirements

Before you optimize Pipeline Manager, be familiar with your existing system configuration, such as:

- Total system memory.
- Other (nonpipeline) processes running on the Pipeline Manager system that will share system resources.
- The number and types of pipelines required for your business logic or planned load balancing.
- The expected load for each pipeline.
- Whether your business logic is more CPU intensive or I/O intensive. (For example, if you use the FCT_Discount module, your business logic is likely to be more CPU intensive.)

Testing Requirements

Before you optimize Pipeline Manager, you should have a set of error-free sample CDRs that resemble those used in your production system.

Optimizing Pipeline Manager

To optimize Pipeline Manager, consider the following actions:

- (Oracle Solaris and Linux) Be sure that OS-specific system configurations were put in place during installation.
- Configure pipelines to run in either single-threaded or multithreaded mode. See "Configuring Single-Threaded or Multithreaded Operation" for more information.

  It is especially important to maximize the performance of the DAT_AccountBatch and DAT_BalanceBatch modules. See:

  – Configuring the DAT_AccountBatch Module Database Connections and Threads
  – Configuring Threads for DAT_BalanceBatch Connections
- Configure function pools within pipelines. See "Optimizing a Pipeline by Using Function Pools" for more information.
- If you have CDR files smaller than a few thousand records, consider grouping multiple CDR files into one transaction. See "Combining Multiple CDR Files into One Transaction" for more information.
- Configure multithreading in the Output Controller. See "Increasing Pipeline Manager Throughput When an EDR Is Associated with Multiple Output Streams" for more information.

- Add additional pipelines. See "Configuring Multiple Pipelines" for more information.
- Verify that any custom iScripts and iRules are efficiently designed. See "Optimizing Function Modules" for more information.
- Configure event and service mapping to only supply the Pipeline Rating Engine with the services being rated.
- Configure the DAT_USC_Map module to improve startup performance.

Troubleshooting Pipeline Performance

Use the following checklist to troubleshoot drops in performance.

- If you installed a patch, find out if the patch changed operating system functions, such as threading or memory management, or made any changes to Pipeline Manager framework modules.
- Check recent customizations, such as iScripts. Look for customizations that might impact database access or hash usage.
- Use database monitoring tools to monitor the Pipeline Manager database to see if there is a lot of activity. If so, check which queries are used and which indexes are used. This might point to the data involved, which might point to the module processing that data.
- Use a monitoring command such as **iostat** to check I/O activity.
- Use a memory monitoring command such as **prstat**, **vmstat**, or **sar** to check if the Pipeline Manager memory usage has changed. If Pipeline Manager uses an unexpected amount of memory, check for duplicate keys related to buffers and call assembly.
- Check for large numbers of files in the following directories:
  - **in**
  - **err**
  - **done**
  - **dupl**
  - **assembl**
  - **rej**

  Delete old files that are no longer needed.
- Look for bottlenecks in the pipeline by using the **prstat** command and the thread ID in the **process.log** file to identify slow threads. Check for:
  - icx (involuntary context switch)
  - vcx (voluntary context switch)
  - scl (system call)
  - slp (sleep)
- Check the **pipeline.log** file for records of a large amount of rollbacks.

Optimizing Function Modules

Slow function modules can be very detrimental to overall Pipeline Manager performance. To optimize individual function modules:

1. Identify the high latency modules by using instrumentation. See "Measuring System Latencies with Instrumentation" for more information.
2. Check if the high latency modules can be optimized. For example, you might discover that the business logic used in high latency iScripts or iRules can be redesigned to improve performance.

Configuring Single-Threaded or Multithreaded Operation

You configure pipelines to run in single-threaded or multithreaded mode by using the **MultiThreaded** registry entry in the registry file.

- **Single-threaded mode:** Use this mode if you are using a system with just a few CPUs and limited RAM.

  In a single-threaded environment, pipelines use a single thread to run all modules and only one CPU is used for each pipeline.

  If the **MultiThreaded** registry entry is not included in the registry file, pipelines will by default run in multithreaded mode.

  > **Note:**
  >
  > Business logic can prevent the setup of multiple pipelines.

- **Multithreaded mode:** Use this mode if your system has many CPUs.

  In a multithreaded environment, pipelines use three or more threads to process each transaction. By default, one thread is used for the input module and one for the output module. An additional thread is used for each function pool that you configure to process function modules.

  For information on optimizing pipelines when using multithreaded mode, see:

  – Assigning Multiple Threads to Process Function Modules
  – Optimizing a Pipeline by Using Function Pools

To configure single-threaded or multithreaded operation:

1. Open the registry file in a text editor.
2. Set the input controller **MultiThreaded** registry entry to the appropriate value:
   - **True** to configure the pipeline for multithreaded processing.
   - **False** to configure the pipeline for single-threaded processing.

```
Pipelines
{
    PipelineName
    {
        MultiThreaded = value
        ...
    }
}
```

3. Restart the pipeline.

Reducing Startup Times with Parallel Loading

You can reduce your startup times by configuring Pipeline Manager to:

- Load all pipelines in parallel.
- Load data modules in parallel.
- Load function modules in parallel.

By default, Pipeline Manager loads pipelines, data modules, and function modules sequentially.

To enable parallel loading, use the Parallel Load Manager module:

1. Open the registry file in a text editor.
2. Configure the **ifw.ParallelLoadManager** section of the registry file:
   - Set the **Active** registry entry to **True**.
   - Set the **NumberOfThreads** registry entry to the number of threads you want Pipeline Manager to use for loading your pipelines, data modules, and function modules.

   For example:

```
ifw
{
   ...
   ParallelLoadManager
   {
      Active = True
      NumberOfThreads = 4
   }
   …
}
```

3. Restart the pipeline.

Assigning Multiple Threads to Process Function Modules

If a pipeline is configured for multithreaded processing and you have idle CPU resources, you might be able to increase performance by grouping function modules into two or more function pools. The pipeline runs each function pool in a separate thread.

> **Note:**
>
> Adding too many function pools to a pipeline can decrease performance because the buffers between the threads consume system CPU overhead and RAM. (Typically, two to six function pools is optimal.)

> **Note:**
>
> If you are using a high-latency module such as FCT_AccountBatch or FCT_Discount and have sufficient hardware resources, assign the module to its own function pool and test for performance improvement.

To create a separate thread for an individual function module or a group of function modules, you use the **FunctionPool** registry entry.

> **Note:**
>
> Before you perform this procedure, read "Optimizing a Pipeline by Using Function Pools" for more information.

1. Submit some sample CDRs to the pipeline with instrumentation enabled. See "Measuring System Latencies with Instrumentation" for more information.
2. Locate the instrumentation results in the **pipeline.log** file.
3. Open the registry file in a text editor.
4. Using the instrumentation data, reduce the processing time required by the slowest function pool by:
   - (Optional) Adding an additional function pool to the **Functions** section of the registry file.
   - Shifting one or more modules from a function pool to an adjacent function pool.

   The objective is to make the processing times of all function pools as similar as possible.
5. Save the registry file.
6. Restart the pipeline.
7. Measure pipeline performance with the sample CDRs by measuring transaction start times and end times in the **pipeline.log** file.
8. Go to Step 3 and repeat testing until optimal results are achieved.

Configuring the DAT_AccountBatch Module Database Connections and Threads

To improve performance, you can configure multiple DAT_AccountBatch connections to the BRM database. Configure the following registry entries:

- Use the **Threads** registry entry to specify the number of threads. Set this value to at least the number of CPUs in the system. Increasing the number of threads increases performance, up to a point. Specifying too many threads decreases performance.

  The default is **4**.
- Use the **Connections** registry entry to specify the number of connections to the database. This value must be at least one more than the number of threads.

  The default is **5**.

- Use the **LoadPercentage** registry entry to specify the percentage of account POIDs to store locally when determining the account blocks for which each thread is responsible.

  Values must be greater than 0.000000 and less than or equal to 100.0.

  The default is **10**.

Setting the Hash Map Size for Threads

You can use the following DAT_AccountBatch registry entries to set the temporary hash map size built for each thread. Each entry controls the hash map size for a different type of data; for example, accounts, logins, and services.

In general, larger maps perform better but consume more memory. Smaller maps save memory but can slow down Pipeline Manager startup. Very low numbers can dramatically slow down Pipeline Manager startup.

The default system-calculated value uses the following formula:

((number of accounts/number of threads) * 2).

The registry entries are:

- **ThreadAccountHashMapSize**: Used for account data.

  > **Note:**
  >
  > Changing the default system-calculated values for this entry is not recommended. Replacing this entry with one larger than the default wastes memory. Replacing this entry with one smaller than the default slows Pipeline Manager startup.

- **ThreadGroupSharingChargesHashMapSize**: Used for charge sharing group data. The system-calculated default value might not be appropriate.

  If your accounts average fewer than two or more than four GroupSharingCharges per account, use the following formula as a guideline to calculate an entry:

  (((number of accounts * average number of GroupSharingCharges per account) / number of threads) * 75%).

- **ThreadGroupSharingDiscountsHashMapSize**: Used for discount sharing group data. The system-calculated default value might not be appropriate.

  If your accounts average fewer than two or more than four GroupSharingDiscounts per account, use the following formula as a guideline to calculate an entry:

  (((number of accounts * average number of GroupSharingDiscounts per account) / number of threads) * 75%).

- **ThreadGroupSharingProfilesHashMapSizes**: Used for profile sharing group data. The system-calculated default value might not be appropriate.

  If your accounts average fewer than two or more than four profile sharing groups per account, use the following formula as a guideline to calculate an entry:

  (((number of accounts * average number of GroupSharingProfiles per account) / number of threads) * 75%).

- **ThreadLoginHashMapSize**: Used for login data. The system-calculated default value is appropriate for most implementations.

If your accounts average more than four logins per account, use the following formula as a guideline to calculate an entry:

(((number of accounts * average number of logins per account) / number of threads) * 75%).

- **ThreadServiceHashMapSize**: Used for service data. The system-calculated default value is appropriate for most implementations.

  If your accounts average more than four services per account, use the following formula as a guideline to calculate an entry:

  (((number of accounts * average number of services per account) / number of threads) * 75%).

Locking Objects during DAT_AccountBatch Processing

You can set the number of pre-allocated mutex objects that are used to lock individual objects during processing to prevent multiple threads from contending for access to the same object. You can use different settings for account, login, and service objects by setting the following DAT_AccountBatch registry entries:

- **AccountLocks**
- **LoginLocks**
- **ServiceLocks**

Usually, the default value for these entries should be appropriate. If you use a larger value, less allocation is needed for additional mutex objects during processing, but more memory is used.

The default for all entries is **10**.

Configuring Threads for DAT_BalanceBatch Connections

Use the following DAT_BalanceBatch registry entry to configure connections to the BRM database:

- **Threads**: Specifies the number of threads for loading the balance data from the BRM database. The number of threads must be smaller than or equal to the number of connections.

  The default is **4**.

- **ThreadHashMapSize**: Specifies the size of the hash map in each thread used for loading balance data from the BRM database.

  The default is **1024**.

Improving Pipeline Manager Startup Performance

For information about improving Pipeline Manager startup performance, see:

- Improving DAT_BalanceBatch Loading Performance
- Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing

Improving DAT_BalanceBatch Loading Performance

The DAT_BalanceBatch module uses the noncurrency balance element validity to select the noncurrency subbalances to load from the BRM database into pipeline memory. If the

noncurrency balance element validity is not configured, at Pipeline Manager startup, DAT_BalanceBatch selects the subbalances that were valid for 366 days by default. When the BRM database contains a large number of noncurrency subbalances, loading them leads to increased Pipeline Manager startup times.

To improve Pipeline Manager startup performance, you can set the noncurrency balance element validity to specify the subbalances to load.

Improving DAT_AccountBatch and DAT_BalanceBatch Load Balancing

The DAT_AccountBatch and DAT_BalanceBatch modules use multithreaded framework to load account and balance data from the BRM database into Pipeline Manager memory. The modules group the accounts and balances into batches or jobs. Multiple worker threads run in parallel to process the jobs. When a thread completes processing, it is assigned another job from the jobs pool, which improves load balancing between the threads and increases Pipeline Manager startup performance.

By default, the number of jobs per thread is 3, which is appropriate in most installations to achieve load balancing. However, if thread loading times vary greatly, you can use the **PerThreadJobsCount** entry in the DAT_AccountBatch registry and the **BalancesPerThreadJobsCount** entry in the DAT_BalanceBatch registry to adjust the number of jobs per thread.

> **Note:**
>
> Setting the number of jobs per thread to a large number can outweigh the performance gain because of the system overhead associated with creating too many jobs. (Typically, three to eight jobs per thread is optimal). To adjust the number of accounts or balances per job, you can do this by increasing or decreasing the number of threads. However, when the number of accounts or balances is too small, the data modules use one thread to optimize performance.

Breaking Up Large Nested Subsections in Registry Files

Pipeline Manager can encounter parser stack overflow errors when a pipeline registry section contains a large number of nested subsections.

You can break up large nested subsections and prevent parser stack overflow errors by using anonymous blocks in your registry file. An anonymous block consists of a nested subsection with braces **{ }** and no subsection name, as shown below:

```
#------------------------------------------------------------------------
# Input section
#------------------------------------------------------------------------
Input
{
   UnitsPerTransaction = 1

   InputModule
   {
      { # <-- Beginning of Anonymous Block
         ModuleName  = INP_GenericStream
         Module
         {
            Grammar = ./formatDesc/Formats/Solution42/SOL42_V670_REL_InGrammar.dsc
```

```
            DefaultOutput = TELOutput

            InputStream
            {
               ModuleName = EXT_InFileManager
               Module
               {
                  InputPath   = ./data/incollect/reprice/in
                  InputPrefix = test_
                  InputSuffix = .edr
                  ...
               }
            } # end of InputStream
         } # end of InputModule
      } # --> End of Anonymous Block
   } # end of InputDataPool
}
```

You can place anonymous blocks in any location and at any hierarchy level of the registry file. For the best effect, divide large sections by placing an anonymous block around a group of smaller subsections. This breaks up the section without affecting the hierarchy of the subsections enclosed within the anonymous block.

Optimizing a Pipeline by Using Function Pools

In general, the performance of a multithreaded pipeline varies directly with its slowest thread. The objective of optimizing a multithreaded pipeline is to group the function modules into function pools so that the slowest function pool is as fast as possible. In this environment, faster threads wait a minimum amount of time for data to be delivered or processed by slower threads.

> **Note:**
>
> Adding too many function pools to a pipeline can decrease performance because the buffers between the threads consume system CPU overhead and RAM. (Typically, two to six function pools is optimal.)

You use instrumentation results to guide function pool configuration. Instrumentation results indicate how many microseconds are required by each module to process a given number of requests. You use this information to add function pools or regroup the modules in existing function pools.

> **Note:**
>
> You cannot improve performance by adding function pools or shifting function modules to adjacent function pools if your *slowest* function pool:
>
> - Has one function module in it. (or)
> - Is faster than the input or output module.
>
> You might be able to improve performance by *reducing* the number of function pools as long as the slowest function pool is *faster* than the output module. (Any performance gain comes from the reduced number of buffers. Fewer buffers require less system process overhead.)

Adding Function Pools

You might improve system performance by adding one or more function pools.

The following example shows a high-level schema of a portion of a registry file for a pipeline called ALL_RATE:

> **Note:**
>
> For information on the buffers between the function pools, see "Configuring Buffers" for more information.

```
input {...}

Functions
{
    PreProcessing
    {
        FunctionPool
        {
            module_1 {}
            module_2 {}
            module_3 {}
        }
    }

    Buffer1 {...}

    Rating
    {
        FunctionPool
        {
            module_4 {}
            module_5 {}
            module_6 {}
        }
    }

    Buffer2 {...}

    PostRating
```

```
    {
        FunctionPool
        {
            module_7 {}
            module_8 {}
        }
    }

output {...}
```

The instrumentation output in the **pipeline.log** file reveals the following latencies for each module for processing a fixed set of test transactions:

> **Note:**
>
> For simplicity, the sample latencies have been rounded to the nearest 5,000,000 microseconds.

```
15.03.2004 13:25:07 testserver ifw IFW NORMAL   00516 -
(ifw.Pipelines.ALL_RATE.Functions.PreProcessing)
Plugin processing time statistics: '
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.module_1.Module, 40000000
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.module_2.Module, 15000000
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.module_3.Module, 45000000

15.03.2004 13:25:07 testserver ifw IFW NORMAL 00516 -
(ifw.Pipelines.ALL_RATE.Functions.Rating)
Plugin processing time statistics: '
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.module_4.Module, 65000000
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.module_5.Module, 30000000
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.module_6.Module, 90000000

15.03.2004 13:25:07 testserver ifw IFW NORMAL 00516 -
(ifw.Pipelines.ALL_RATE.Functions.PostRating)
Plugin processing time statistics: '
ifw.Pipelines.ALL_RATE.Functions.Postrating.FunctionPool.module_7.Module, 35000000
ifw.Pipelines.ALL_RATE.Functions.Postrating.FunctionPool.module_8.Module, 50000000
```

This output is summarized in Table 71-1:

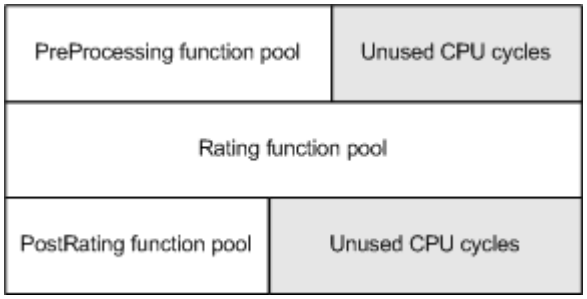
Table 71-1 Example 1 Module Latencies Summary

Module	Module Latency (Microseconds)	Function Pool	Function Pool Latency (Microseconds)
module_1	40,000,000	PreProcessing	100,000,000
module_2	15,000,000	PreProcessing	100,000,000
module_3	45,000,000	PreProcessing	100,000,000
module_4	65,000,000	Rating	185,000,000
module_5	30,000,000	Rating	185,000,000
module_6	90,000,000	Rating	185,000,000
module_7	35,000,000	PostRating	85,000,000
module_8	50,000,000	PostRating	85,000,000

The total latency in this configuration is 185,000,000; this represents the microseconds used by the slowest function pool.

Figure 71-1 shows that about a third of the CPU cycles used by the function pool threads are idle:

Figure 71-1 Unused CPU Cycles Example 1



In this example, the pipeline can be optimized if module_6 is assigned to its own function pool, as in this revised sample:

```
input {...}

Functions
{
    PreProcessing
    {
        FunctionPool
        {
            module_1 {}
            module_2 {}
            module_3 {}
        }
    }

    Buffer1 {...}

    Rating
    {
        FunctionPool
        {
            module_4 {}
            module_5 {}
         }

    Buffer2 {...}

    Discounting
    {
        functionpool
        {
            module_6 {}
        }
    }

    Buffer3 {...}

    PostRating
    {
```

```
        FunctionPool
        {
            module_7 {}
            module_8 {}
        }
    }

output {...}
```

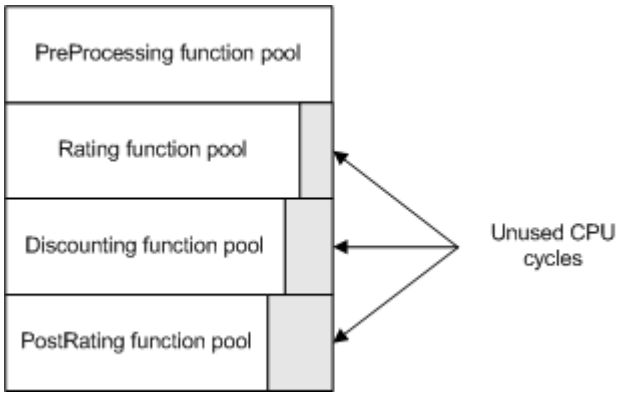
The latency table now appears as shown in Table 71-2:

Table 71-2 Example 2 Modules Latencies Summary

Module	Module Latency (Microseconds)	Function Pool	Function Pool Latency (Microseconds)
module_1	40,000,000	PreProcessing	100,000,000
module_2	15,000,000	PreProcessing	100,000,000
module_3	45,000,000	PreProcessing	100,000,000
module_4	65,000,000	Rating	95,000,000
module_5	30,000,000	Rating	95,000,000
module_6	90,000,000	Discounting	90,000,000
module_7	35,000,000	PostRating	85,000,000
module_8	50,000,000	PostRating	85,000,000

Total function module latency in the new configuration is 100,000,000 microseconds, equivalent to the latency of the PreProcessing function pool. Less than eight percent of function pool CPU cycles are now idle as shown by the gray cycles in Figure 71-2:

Figure 71-2 Unused CPU Cycles Example 2



Shifting Modules between Function Pools

Adding an additional function pool can decrease performance in some situations (see "Adding Function Pools" for more information). This can occur if the system overhead for the additional buffer more than offsets the performance gains from a faster highest-latency function pool. When this occurs, you might be able to improve performance by keeping the number of function pools constant and shifting modules to adjoining function pools.

In the sample above, if adding an additional function pool decreased performance, you could return to using three function pools and then move module 4 to the end of the PreProcessing function pool as shown in Table 71-3:

Table 71-3 Example 3 Modules Latencies Summary

Module	Module Latency (Microseconds)	Function Pool	Function Pool Latency (Microseconds)
module_1	40,000,000	PreProcessing	165,000,000
module_2	15,000,000	PreProcessing	165,000,000
module_3	45,000,000	PreProcessing	165,000,000
module_4	65,000,000	PreProcessing	165,000,000
module_5	30,000,000	Rating	120,000,000
module_6	90,000,000	Rating	120,000,000
module_7	35,000,000	PostRating	85,000,000
module_8	50,000,000	PostRating	85,000,000

Total function module latency in the new configuration is 165,000,000 microseconds. This is equivalent to the latency of the PreProcessing function pool. Though performance gains might be more modest than in the first scenario (where a new function pool was added), the performance gain is more certain because no additional buffer overhead was added.

Configuring Buffers

In a multithreaded pipeline, each pair of consecutive threads communicates through a buffer. Because each function pool is assigned a thread, you must configure a buffer between consecutive function pools.

You configure the buffers between function pool sections in the pipeline registry file. Normally, each buffer can be configured as follows:

```
Buffer1
        {
          Size = 100
        }
```

> **Note:**
>
> On Solaris systems, you should configure block transfer mode.

Combining Multiple CDR Files into One Transaction

Pipeline Manager is generally more efficient when it processes large CDR files. If a pipeline receives and processes small CDR files, you can improve processing performance by combining multiple CDR input files into one pipeline transaction. You use the **UnitsPerTransaction** registry entry in the input controller to implement this functionality.

The **UnitsPerTransaction** entry specifies the number of CDR input files that make up a transaction. By default, each CDR file forms its own transaction.

> **Note:**
>
> The optimal transaction size depends on your system configuration and pricing model. In general, most system configurations perform best when the total number of CDRs, which is the average number of CDRs per input file multiplied by the number of input files in the transaction, is greater than 10,000.

If the **UnitsPerTransaction** value is greater than **1**, you can use the **SequenceGeneration** registry entry in the output controller to specify whether the pipeline generates one output file per CDR input file or one output file for the entire transaction. Pipeline Manager performance is generally faster when one output file is generated for the entire (multi-CDR) transaction.

To combine multiple CDR files into one transaction:

1. In the **Input** section of the registry file, set the **UnitsPerTransaction** entry to the number of CDR input files that make up one transaction. For example, set **UnitsPerTransaction** to **100** to combine 100 CDR input files into one transaction.

   > **Note:**
   >
   > The default **UnitsPerTransaction** value is **1**.

   ```
   Input
   {
       ...
       UnitsPerTransaction = 100
       ...
   }
   ```

2. (Optional) In the **Output** section of the registry file, set the **SequenceGeneration** entry to **Transaction**. This configures the pipeline to generate one output file for the entire transaction.

   > **Note:**
   >
   > The default **SequenceGeneration** value is **Units**, which configures the pipeline to generate one output file per CDR input file.

   ```
   Output
   {
       ...
       SequenceGeneration = Transaction
       ...
   }
   ```

3. Stop and restart the pipeline.

Increasing Pipeline Manager Throughput When an EDR Is Associated with Multiple Output Streams

You can enhance Pipeline Manager throughput by configuring multithreading in the Output Controller. This enables Pipeline Manager to write multiple EDRs in parallel when the EDRs are associated with multiple output streams.

> **Note:**
>
> Enable multithreading in the Output Controller only if the EDRs are associated with multiple output streams.

Enabling multithreading may cause an increase in the overall memory usage of the Output Controller. However, the memory usage becomes constant after processing EDRs for some time.

To configure multithreading in the Output Controller:

1. Open the registry file (for example, *pipeline_home***/conf/wireless.reg**) in a text editor.
2. In the **MultiThreading** section, do the following:
   - Set the **Active** registry entry to **True**.
   - Set the **NumberOfThreads** registry entry to the number of threads you want the Output Controller to create for Pipeline Manager to write multiple EDRs in parallel.
   - Set the **BatchSize** registry entry to the appropriate value:
     - **0** indicates that the Output Controller does not run in batch mode.
     - A value greater than **0** indicates that the Output Controller operates in batch mode with the batch size equal to the specified value.

   For example:

```
Output
{
   ...
   ...
   MultiThreading
   {
       Active = True
       NumberOfThreads = 5
       BatchSize = 500
    }
}
```

3. Save and close the file.
4. Restart the pipeline.

Configuring Multiple Pipelines

If you have high transaction throughput requirements and additional system balance elements, you might improve system performance by running multiple pipelines that perform the same function.

In general, consider running multiple pipelines if:

- Your system has a relatively large number of CPUs.
- The order of the input streams is not important.

> **Note:**
>
> When you use the FCT_CallAssembling or the FCT_DuplicateCheck module, you must process the EDRs for the same account in the same pipeline.

> **Note:**
>
> If you configure multiple pipelines and your system is running at near full capacity on a limited number of CPUs, test running the pipelines in single-threaded mode. This configuration reduces the buffer memory allocation requirement and thread-handling overhead. To enable single-threaded operation, set the **MultiThreaded** entry to **False**. See "Assigning Multiple Threads to Process Function Modules" for more information.

Customizing Flists Sent to a Real-Time Pipeline

You can configure the fields included in flists sent to a real-time pipeline by using the **load_pin_rtp_trim_flist** utility. This utility is useful for:

- Improving system efficiency by removing (trimming) fields that the pipeline does not use.
- Supporting custom iScripts and iRules in the pipeline by adding fields to default flists that are not included in the flists by default.

To optimize the set of fields sent to a real-time pipeline:

1. Determine which fields are required by the real-time pipeline.
2. Create an XML file that describes the fields to be sent to the real-time pipeline based on one or more event types.
3. Load the XML file using the **load_pin_rtp_trim_flist** utility.

Configuration Object Dot Notation

The **load_pin_rtp_trim_flist** utility creates a configuration object (**/config/rtp/trim_flist**). This object is used to create the trimmed flists.

The configuration object uses dot notation. For example, the PIN_FLD_STATUS_FLAGS field at the end of this portion of a sample flist:

```
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 32, used 32
1     PIN_FLD_POID           POID [0] 0.0.0.1 /account 10243 13
1     PIN_FLD_MOD_T        TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
1     PIN_FLD_ACCOUNT_NO      STR [0] "0.0.0.1-10243"
1     PIN_FLD_CURRENCY        INT [0] 840
1     PIN_FLD_BILL_WHEN       INT [0] 1
1     PIN_FLD_LAST_BILL_T  TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_SERVICE_INFO   SUBSTRUCT [0] allocated 32, used 32
2         PIN_FLD_STATUS         ENUM [0] 10100
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
```

is represented as:

```
PIN_FLD_INHERITED_INFO.PIN_FLD_SERVICE_INFO.PIN_FLD_STATUS_FLAGS
```

in the configuration object.

About the *field_list*.xml File

The *field_list***.xml** file specifies the fields from the **/account** and **/service** objects that are included in the flist that is sent to Pipeline Manager. You can define conditions in **<EventMap>** sections in the XML file that indicate which fields should be included in the flist depending on the event type.

The following example shows the XML file structure with session and provisioning event filters:

```
<EventMapList>

  <!--* The following event map specifies fields sent
      * when the event type is exactly /event/session -->

  <EventMap>
    <Event>
      <Type>/event/session</Type>
      <Flags>0</Flags>
    </Event>
    <RequiredField>

        <!-- List of fields sent put here. -->

    </RequiredField>
  </EventMap>

  <!--* The following event map specifies fields sent
      * when the event type starts with /event/session/ -->

  <EventMap>
    <Event>
      <Type>/event/session/</Type>
      <Flags>1</Flags>
    </Event>
    <RequiredField>

        <!-- List of fields sent put here. -->

    </RequiredField>
  </EventMap>

  <!--* The following event map specifies fields sent
      * when when a provisioning event matches any of three conditions. -->
```

```
<EventMap>
  <Event>
    <Type>/event/provisioning</Type>
    <Flags>0</Flags>
  </Event>
  <Event>
    <Type>/event/provisioning/session</Type>
    <Flags>0</Flags>
  </Event>
  <Event>
    <Type>/event/provisioning/</Type>
    <Flags>1</Flags>
  </Event>
  <RequiredField>

      <!-- List of fields sent put here. -->

  </RequiredField>
</EventMap>

<!--* The following event map specifies fields sent when none of the
    * above conditions are true. -->

<EventMap>
  <Event>
    <Type>*</Type>
    <Flags>1</Flags>
  </Event>
  <RequiredField>

      <!-- List of fields sent put here. -->

  </RequiredField>
</EventMap>
</EventMapList>
```

The **Flags** tag in the XML file specifies event matching criteria.

- A **Flags** value of **0** specifies that an exact match is required.
- A **Flags** value of **1** specifies that the event type must start with the string specified in the **Type** tag. The value **1** is also used when indicating **Type** value asterisk (*****). This value matches all event types.

> **Note:**
>
> Search order is important. The fields included with the flist are the fields specified in the first event map section of the XML file where the event type matches the string in the **Type** field.

You can use the sample XML fields list (*BRM_home***/sys/data/config/pin_config_rtp_trim_flist.xml)** as a base for your custom XML file.

For a detailed example using session event filters, see "Usage Example".

Mapping Events to Flists

Because one flist can be used by more than one event, you can specify the relationship between an event and the flist.

For example, the following section is of an event map XML file:

```
<EventMap>
      <Event>
        <Type>/event/session</Type>
        <Flags>0</Flags>
      </Event>
      <Event>
        <Type>/event/session/</Type>
        <Flags>1</Flags>
      </Event>
```

is mapped to an flist as follows:

```
0 PIN_FLD_EVENT_MAP              ARRAY [0] allocated 20, used 8
1     PIN_FLD_EVENTS             ARRAY [0] allocated 20, used 8
2         PIN_FLD_EVENT_TYPE           STR [0] "/event/session"
2         PIN_FLD_FLAGS                INT [0] 0
1     PIN_FLD_EVENTS             ARRAY [1] allocated 20, used 8
2         PIN_FLD_EVENT_TYPE           STR [0] "/event/session/"
2         PIN_FLD_FLAGS                INT [0] 1
```

Usage Example

An unmodified flist might look like the sample shown in "Sample Unmodified Flist". However, in this example, Pipeline Manager only requires subsets of fields listed in "Sample Fields Required by Pipeline Manager" depending on the event type.

In this example, to implement the trimmed flist:

1. Create the XML file shown in "sample.xml File" to modify the default list of fields ("Sample Unmodified Flist") included in the flist.
2. Load the XML file using the utility:

   load_pin_rtp_trim_flist -f sample.xml [**-v**] [**-d**]

Sample Unmodified Flist

The following is the default (untrimmed) list of fields sent to Pipeline Manager.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /event/session -1 0
0 PIN_FLD_EVENT        SUBSTRUCT [0] allocated 25, used 25
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/session -1 0
1     PIN_FLD_NAME            STR [0] "Activity Session Log"
1     PIN_FLD_USERID         POID [0] 0.0.0.1 /service/pcm_client 1 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 0
1     PIN_FLD_PROGRAM_NAME    STR [0] "testnap"
1     PIN_FLD_START_T      TSTAMP [0] (1065785673) Fri Oct 10 04:34:33 2003
1     PIN_FLD_END_T        TSTAMP [0] (1065785683) Fri Oct 10 04:34:43 2003
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip 11907 1
1     PIN_FLD_SYS_DESCR       STR [0] "Session: generic"
1     PIN_FLD_RUM_NAME        STR [0] "Duration"
1     PIN_FLD_UNIT           ENUM [0] 1
```

```
1     PIN_FLD_TOD_MODE       ENUM [0] 2
1     PIN_FLD_NET_QUANTITY DECIMAL [0] 60.000000000000000
1     PIN_FLD_MIN_QUANTITY DECIMAL [0] 60.000000000000000
1     PIN_FLD_INCR_QUANTITY DECIMAL [0] 60.000000000000000
1     PIN_FLD_MIN_UNIT       ENUM [0] 2
1     PIN_FLD_INCR_UNIT      ENUM [0] 2
1     PIN_FLD_ROUNDING_MODE   ENUM [0] 1
1     PIN_FLD_TIMEZONE_MODE   ENUM [0] 1
1     PIN_FLD_RATED_TIMEZONE_ID    STR [0] "GMT-08:00"
1     PIN_FLD_TIMEZONE_ADJ_START_T TSTAMP [0] (1065760473) Thu Oct 09 21:34:33 2003
1     PIN_FLD_TIMEZONE_ADJ_END_T TSTAMP [0] (1065760483) Thu Oct 09 21:34:43 2003
1     PIN_FLD_TOTAL         ARRAY [840] allocated 20, used 1
2         PIN_FLD_AMOUNT       DECIMAL [0] 0.0166667
1     PIN_FLD_BAL_IMPACTS   ARRAY [0] allocated 20, used 17
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 13
2         PIN_FLD_AMOUNT       DECIMAL [0] 0.0166667
2         PIN_FLD_RESOURCE_ID     INT [0] 840
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 10030 0
2         PIN_FLD_RATE_OBJ       POID [0] 0.0.0.1 /rate 9390 1
2         PIN_FLD_DISCOUNT     DECIMAL [0] 0
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_GL_ID           INT [0] 104
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 1
2         PIN_FLD_QUANTITY     DECIMAL [0] 60.00000000
2         PIN_FLD_RATE_TAG        STR [0] "$1 per hour"
2         PIN_FLD_TAX_CODE        STR [0] ""
2         PIN_FLD_IMPACT_CATEGORY    STR [0] "default"
2         PIN_FLD_PACKAGE_ID    INT [0] 20030910
2         PIN_FLD_LINEAGE         STR [0] ""
2         PIN_FLD_PERCENT      DECIMAL [0] 1
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_UNRATED_QUANTITY DECIMAL [0] 0
0 PIN_FLD_DISCOUNTS     ARRAY [0] allocated 20, used 8
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 0
1     PIN_FLD_OWNER_OBJ      POID [0] 0.0.0.1 /service/ip 11907 1
1     PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_DISCOUNT_LIST  ARRAY [0] allocated 20, used 19
2         PIN_FLD_CREATED_T    TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0  0 0
2         PIN_FLD_DESCR           STR [0] ""
2         PIN_FLD_DISCOUNT_OBJ   POID [0] 0.0.0.1 /discount 8273 0
2         PIN_FLD_LAST_MODIFIED_T TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2         PIN_FLD_PACKAGE_ID    INT [0] 12222
2         PIN_FLD_PLAN_OBJ       POID [0] 0.0.0.0  0 0
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.0  0 0
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 1
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_FLAGS           INT [0] 1
2         PIN_FLD_TYPE           ENUM [0] 602
1     PIN_FLD_DISCOUNT_LIST  ARRAY [1] allocated 20, used 19
2         PIN_FLD_CREATED_T    TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (1071385462) Sat Dec 13 23:04:22 2003
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (1052895862) Wed May 14 00:04:22 2003
2         PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.0  0 0
2         PIN_FLD_DESCR           STR [0] ""
```

```
2         PIN_FLD_DISCOUNT_OBJ   POID [0] 0.0.0.1 /discount 11345 0
2         PIN_FLD_LAST_MODIFIED_T TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
2         PIN_FLD_PACKAGE_ID    INT [0] 22222
2         PIN_FLD_PLAN_OBJ       POID [0] 0.0.0.0  0 0
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (1068793462) Thu Nov 13 23:04:22 2003
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/ip 11907 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 1
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (1068793462) Thu Nov 13 23:04:22 2003
2         PIN_FLD_USAGE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_FLAGS           INT [0] 1
2         PIN_FLD_TYPE           ENUM [0] 602
1     PIN_FLD_DISCOUNT_LIST  ARRAY [2] allocated 28, used 28
2         PIN_FLD_POID           POID [0] 0.0.0.1 /discount 8219 1
2         PIN_FLD_CREATED_T    TSTAMP [0] (1064333980) Tue Sep 23 09:19:40 2003
2         PIN_FLD_MOD_T        TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_READ_ACCESS     STR [0] "B"
2         PIN_FLD_WRITE_ACCESS    STR [0] "S"
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 1
2         PIN_FLD_DESCR           STR [0] ""
2         PIN_FLD_END_T        TSTAMP [0] (1069333980) Thu Nov 20 05:13:00 2003
2         PIN_FLD_MODE           ENUM [0] 801
2         PIN_FLD_NAME            STR [0] "System discount 1"
2         PIN_FLD_OWN_MAX      DECIMAL [0] 0
2         PIN_FLD_OWN_MIN      DECIMAL [0] 0
2         PIN_FLD_PERMITTED       STR [0] ""
2         PIN_FLD_PRIORITY     DECIMAL [0] 1
2         PIN_FLD_PURCHASE_MAX DECIMAL [0] 0
2         PIN_FLD_PURCHASE_MIN DECIMAL [0] 0
2         PIN_FLD_START_T      TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_TYPE           ENUM [0] 603
2         PIN_FLD_USAGE_MAP     ARRAY [0] allocated 20, used 4
3             PIN_FLD_DISCOUNT_MODEL    STR [0] "DMStandard"
3             PIN_FLD_EVENT_TYPE      STR [0] "/event"
3             PIN_FLD_FLAGS           INT [0] 0
3             PIN_FLD_SNOWBALL_FLAG    INT [0] 0
2         PIN_FLD_DISCOUNT_OBJ   POID [0] 0.0.0.1 /discount 8219 1
2         PIN_FLD_SERVICE_OBJ    POID [0] NULL poid pointer
2         PIN_FLD_PACKAGE_ID    INT [0]
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_USAGE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (1069333980) Thu Nov 20 05:13:00 2003
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (1069333980) Thu Nov 20 05:13:00 2003
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_FLAGS           INT [0] 1
1     PIN_FLD_DISCOUNT_LIST  ARRAY [3] allocated 28, used 28
2         PIN_FLD_POID           POID [0] 0.0.0.1 /discount 9755 1
2         PIN_FLD_CREATED_T    TSTAMP [0] (1064334036) Tue Sep 23 09:20:36 2003
2         PIN_FLD_MOD_T        TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_READ_ACCESS     STR [0] "B"
2         PIN_FLD_WRITE_ACCESS    STR [0] "S"
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 1
2         PIN_FLD_DESCR           STR [0] ""
2         PIN_FLD_END_T        TSTAMP [0] (1069334036) Thu Nov 20 05:13:56 2003
2         PIN_FLD_MODE           ENUM [0] 801
2         PIN_FLD_NAME            STR [0] "Sys discount 3"
2         PIN_FLD_OWN_MAX      DECIMAL [0] 0
2         PIN_FLD_OWN_MIN      DECIMAL [0] 0
2         PIN_FLD_PERMITTED       STR [0] ""
2         PIN_FLD_PRIORITY     DECIMAL [0] 14
```

```
2         PIN_FLD_PURCHASE_MAX DECIMAL [0] 0
2         PIN_FLD_PURCHASE_MIN DECIMAL [0] 0
2         PIN_FLD_START_T      TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_TYPE           ENUM [0] 603
2         PIN_FLD_USAGE_MAP     ARRAY [0] allocated 20, used 4
3             PIN_FLD_DISCOUNT_MODEL    STR [0] "DMStandard"
3             PIN_FLD_EVENT_TYPE      STR [0] "/event/session"
3             PIN_FLD_FLAGS           INT [0] 1
3             PIN_FLD_SNOWBALL_FLAG    INT [0] 0
2         PIN_FLD_DISCOUNT_OBJ   POID [0] 0.0.0.1 /discount 9755 1
2         PIN_FLD_SERVICE_OBJ    POID [0] NULL poid pointer
2         PIN_FLD_PACKAGE_ID    INT [0]
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_USAGE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (1069334036) Thu Nov 20 05:13:56 2003
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (1069334036) Thu Nov 20 05:13:56 2003
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_FLAGS           INT [0] 1
1     PIN_FLD_DISCOUNT_LIST  ARRAY [4] allocated 28, used 28
2         PIN_FLD_POID           POID [0] 0.0.0.1 /discount 11291 1
2         PIN_FLD_CREATED_T    TSTAMP [0] (1064334029) Tue Sep 23 09:20:29 2003
2         PIN_FLD_MOD_T        TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_READ_ACCESS     STR [0] "B"
2         PIN_FLD_WRITE_ACCESS    STR [0] "S"
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 1
2         PIN_FLD_DESCR           STR [0] ""
2         PIN_FLD_END_T        TSTAMP [0] (1069334029) Thu Nov 20 05:13:49 2003
2         PIN_FLD_MODE           ENUM [0] 801
2         PIN_FLD_NAME            STR [0] "Sys discount 2"
2         PIN_FLD_OWN_MAX      DECIMAL [0] 0
2         PIN_FLD_OWN_MIN      DECIMAL [0] 0
2         PIN_FLD_PERMITTED       STR [0] ""
2         PIN_FLD_PRIORITY     DECIMAL [0] 200
2         PIN_FLD_PURCHASE_MAX DECIMAL [0] 0
2         PIN_FLD_PURCHASE_MIN DECIMAL [0] 0
2         PIN_FLD_START_T      TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_TYPE           ENUM [0] 603
2         PIN_FLD_USAGE_MAP     ARRAY [0] allocated 20, used 4
3             PIN_FLD_DISCOUNT_MODEL    STR [0] "DMStandard"
3             PIN_FLD_EVENT_TYPE      STR [0] "/event/session"
3             PIN_FLD_FLAGS           INT [0] 1
3             PIN_FLD_SNOWBALL_FLAG    INT [0] 0
2         PIN_FLD_DISCOUNT_OBJ   POID [0] 0.0.0.1 /discount 11291 1
2         PIN_FLD_SERVICE_OBJ    POID [0] NULL poid pointer
2         PIN_FLD_PACKAGE_ID    STR [0]
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_USAGE_START_T TSTAMP [0] (1061399955) Wed Aug 20 10:19:15 2003
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (1069334029) Thu Nov 20 05:13:49 2003
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (1069334029) Thu Nov 20 05:13:49 2003
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_FLAGS           INT [0] 1
0 PIN_FLD_BAL_INFO      ARRAY [0] allocated 20, used 3
1     PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_BALANCES      ARRAY [840] allocated 11, used 6
2         PIN_FLD_NEXT_BAL     DECIMAL [0] 0
2         PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
2         PIN_FLD_CURRENT_BAL  DECIMAL [0] 19.590836
2         PIN_FLD_CREDIT_LIMIT DECIMAL [0] 100
2         PIN_FLD_CREDIT_FLOOR DECIMAL [0] 0
2         PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
1     PIN_FLD_BALANCES      ARRAY [1000001] allocated 7, used 6
2         PIN_FLD_NEXT_BAL     DECIMAL [0] 0
```

```
2         PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
2         PIN_FLD_CURRENT_BAL  DECIMAL [0] 0
2         PIN_FLD_CREDIT_LIMIT DECIMAL [0] 100
2         PIN_FLD_CREDIT_FLOOR DECIMAL [0] 0
2         PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 32, used 32
1     PIN_FLD_POID           POID [0] 0.0.0.1 /account 10243 13
1     PIN_FLD_MOD_T        TSTAMP [0] (1063218065) Wed Sep 10 11:21:05 2003
1     PIN_FLD_ACCOUNT_NO      STR [0] "0.0.0.1-10243"
1     PIN_FLD_BRAND_OBJ      POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_TIMEZONE_ID     STR [0] ""
1     PIN_FLD_STATUS         ENUM [0] 10100
1     PIN_FLD_STATUS_FLAGS    INT [0] 0
1     PIN_FLD_CURRENCY        INT [0] 840
1     PIN_FLD_CURRENCY_SECONDARY    INT [0] 0
1     PIN_FLD_GROUP_OBJ      POID [0] 0.0.0.0  0 0
1     PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
1     PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 8835 0"
1     PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
1     PIN_FLD_ACTG_TYPE      ENUM [0] 2
1     PIN_FLD_LAST_STATUS_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_GL_SEGMENT      STR [0] "."
1     PIN_FLD_BILL_WHEN       INT [0] 1
1     PIN_FLD_PAY_TYPE       ENUM [0] 10001
1     PIN_FLD_AR_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 8451 0
1     PIN_FLD_NEXT_BILL_OBJ   POID [0] 0.0.0.0  0 0
1     PIN_FLD_NEXT_BILL_T  TSTAMP [0] (1065769200) Fri Oct 10 00:00:00 2003
1     PIN_FLD_LAST_BILL_T  TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_ACTG_LAST_T  TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
1     PIN_FLD_ACTG_FUTURE_T TSTAMP [0] (1068451200) Mon Nov 10 00:00:00 2003
1     PIN_FLD_BILL_ACTGCYCLES_LEFT    INT [0] 1
1     PIN_FLD_PAYINFO_OBJ    POID [0] 0.0.0.1 /payinfo/invoice 11267 0
1     PIN_FLD_ACTG_NEXT_T  TSTAMP [0] (1065769200) Fri Oct 10 00:00:00 2003
1     PIN_FLD_LAST_BILL_OBJ   POID [0] 0.0.0.0  0 0
1     PIN_FLD_BILL_OBJ       POID [0] 0.0.0.1 /bill 10499 0
1     PIN_FLD_PENDING_RECV DECIMAL [0] 0
1     PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_SERVICE_INFO SUBSTRUCT [0] allocated 51, used 26
2         PIN_FLD_POID           POID [0] 0.0.0.1 /service/ip 11907 5
2         PIN_FLD_CREATED_T    TSTAMP [0] (1063217471) Wed Sep 10 11:11:11 2003
2         PIN_FLD_MOD_T        TSTAMP [0] (1063217473) Wed Sep 10 11:11:13 2003
2         PIN_FLD_READ_ACCESS     STR [0] "L"
2         PIN_FLD_WRITE_ACCESS    STR [0] "L"
2         PIN_FLD_AAC_ACCESS      STR [0] ""
2         PIN_FLD_AAC_PACKAGE     STR [0] ""
2         PIN_FLD_AAC_PROMO_CODE    STR [0] ""
2         PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
2         PIN_FLD_AAC_SOURCE      STR [0] ""
2         PIN_FLD_AAC_VENDOR      STR [0] ""
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 0
2         PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
2         PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
2         PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/cycle_forward 11651 0"
2         PIN_FLD_LASTSTAT_CMNT    STR [0] ""
2         PIN_FLD_LAST_STATUS_T TSTAMP [0] (1063217469) Wed Sep 10 11:11:09 2003
2         PIN_FLD_LOGIN           STR [0] "00491732411"
2         PIN_FLD_NAME            STR [0] "PIN Service Object"
2         PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
2         PIN_FLD_PASSWD          STR [0] "clear|00491732411"
2         PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
2         PIN_FLD_STATUS         ENUM [0] 10100
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
```

```
2         PIN_FLD_SERVICE_IP   SUBSTRUCT [0] allocated 20, used 3
3             PIN_FLD_COMPRESSION    ENUM [0] 0
3             PIN_FLD_IPADDR       BINSTR [0] 1 00
3             PIN_FLD_PROTOCOL       ENUM [0] 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
```

Sample Fields Required by Pipeline Manager

The following are sample fields, in flist format, required by Pipeline Manager when the event type is **/event/session**:

> **Note:**
>
> You cannot trim the default fields for the PIN_FLD_INHERITED_INFO substruct listed in "Sample Unmodified Flist". However, you can specify additional **/account** and **/service** fields. In the text below, the **/account** field PIN_FLD_RESIDENCE_FLAG is specified at the end of the list. It is added to the default PIN_FLD_INHERITED_INFO fields sent to Pipeline Manager.

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /event/session -1 0
0 PIN_FLD_EVENT        SUBSTRUCT [0] allocated 25, used 25
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/session -1 0
1     PIN_FLD_START_T      TSTAMP [0] (1065785673) Fri Oct 10 04:34:33 2003
1     PIN_FLD_END_T        TSTAMP [0] (1065785683) Fri Oct 10 04:34:43 2003
1     PIN_FLD_BAL_IMPACTS   ARRAY [0] allocated 20, used 17 and other array elements
2         PIN_FLD_AMOUNT       DECIMAL [0] 0.0166667
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_RESOURCE_ID     INT [0] 840
2         PIN_FLD_GL_ID           INT [0] 104
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 1
2         PIN_FLD_QUANTITY     DECIMAL [0] 60.00000000
2         PIN_FLD_RATE_TAG        STR [0] "$1 per hour"
2         PIN_FLD_TAX_CODE        STR [0] ""
0 PIN_FLD_DISCOUNTS     ARRAY [0] allocated 20, used 8 and other array elements
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 10243 0
1     PIN_FLD_OWNER_OBJ      POID [0] 0.0.0.1 /service/ip 11907 1
1     PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_DISCOUNT_LIST  ARRAY [0] allocated 20, used 19 and other array elements
2         PIN_FLD_DISCOUNT_OBJ   POID [0] 0.0.0.1 /discount 8273 0
2         PIN_FLD_PACKAGE_ID    INT [0] 12222
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_T TSTAMP [0] (1052871608) Tue May 13 17:20:08 2003
2         PIN_FLD_FLAGS           INT [0] 1
2         PIN_FLD_TYPE           ENUM [0] 602
0 PIN_FLD_BAL_INFO      ARRAY [0] allocated 20, used 3 and other array elements
1     PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 8323 4
1     PIN_FLD_BALANCES      ARRAY [840] allocated 11, used 6 and other array elements
2         PIN_FLD_CURRENT_BAL  DECIMAL [0] 19.590836
1     PIN_FLD_BALANCES      ARRAY [1000001] allocated 7, used 6 and other array elements
2         PIN_FLD_CURRENT_BAL  DECIMAL [0] 0
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 32, used 32
1     PIN_FLD_RESIDENCE_FLAG ENUM [0] 1
```

A different set of fields is required when the event type is **/event/session/** (including the final forward slash), and another set of fields is sent for any other type of event.

To implement the trimmed flist in the example, create the following XML file (**sample.xml**). When this XML file is loaded with **load_pin_rtp_trim_flist**, the flist sent to Pipeline Manager is constructed as follows:

- If the event type is exactly **/event/session**, the PIN_FLD_RESIDENCE_FLAG field is included with the trimmed flist as shown in the flist sample above.
- If the event type starts with **/event/session/** (including the last forward slash), the PIN_FLD_RESIDENCE_FLAG field is not included with the trimmed flist.
- If the event type is any other value (which matches the section specified by **Type** value ***** with **Flags** value **1**), then neither the PIN_FLD_RESIDENCE_FLAG field nor the PIN_FLD_BAL_IMPACTS array is included with the trimmed flist.

> **Note:**
>
> You cannot trim the default fields for the PIN_FLD_INHERITED_INFO substruct listed in "Sample Unmodified Flist". However, you can specify additional **/account** and **/service** fields. In the text below, the **/account** field PIN_FLD_RESIDENCE_FLAG is specified at the end of the list. It is added to the default PIN_FLD_INHERITED_INFO fields sent to Pipeline Manager.

sample.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
================================================================
Copyright (c) 2004 Portal Software, Inc. All rights reserved.
This material is the confidential property of Portal Software, Inc.
or its Subsidiaries or licensors and may be used, reproduced, stored
or transmitted only in accordance with a valid Portal license or
sublicense agreement.
================================================================
-->

<RTPTrimFlistConfiguration xmlns="http://www.portal.com/InfranetXMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.portal.com/InfranetXMLSchema pin_config_rtp_trim_flist.xsd">
  <EventMapList>
    <EventMap>

<!-- Section which specifies fields sent when the event type is exactly /event/session -->

      <Event>
        <Type>/event/session</Type>
        <Flags>0</Flags>
      </Event>
      <RequiredField>
        <Name>PIN_FLD_POID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_EVENT</Name>
        <RequiredField>
          <Name>PIN_FLD_POID</Name>
        </RequiredField>
```

```
    <RequiredField>
      <Name>PIN_FLD_START_T</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_END_T</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_BAL_IMPACTS</Name>
      <RequiredField>
        <Name>PIN_FLD_AMOUNT</Name>
      </RequiredField>
        <RequiredField>
        <Name>PIN_FLD_AMOUNT_DEFERRED</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_RESOURCE_ID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_GL_ID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_IMPACT_TYPE</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_QUANTITY</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_RATE_TAG</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_TAX_CODE</Name>
      </RequiredField>
    </RequiredField>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_DISCOUNTS</Name>
    <RequiredField>
      <Name>PIN_FLD_ACCOUNT_OBJ</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_OWNER_OBJ</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_BAL_GRP_OBJ</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_DISCOUNT_LIST</Name>
      <RequiredField>
        <Name>PIN_FLD_DISCOUNT_OBJ</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_PACKAGE_ID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_PURCHASE_END_T</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_PURCHASE_START_T</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_QUANTITY</Name>
      </RequiredField>
```

```
          <RequiredField>
            <Name>PIN_FLD_STATUS</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_USAGE_END_T</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_USAGE_START_T</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_FLAGS</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_TYPE</Name>
          </RequiredField>
        </RequiredField>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_BAL_INFO</Name>
        <RequiredField>
          <Name>PIN_FLD_BAL_GRP_OBJ</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_BALANCES</Name>
          <RequiredField>
            <Name>PIN_FLD_CURRENT_BAL</Name>
          </RequiredField>
        </RequiredField>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_INHERITED_INFO</Name>
        <RequiredField>
          <Name>PIN_FLD_RESIDENCE_FLAG</Name>
        </RequiredField>
      </RequiredField>
    </EventMap>

<!-- Section which specifies fields sent when the event type starts with /event/session/ -->

    </EventMap>
      <Event>
        <Type>/event/session/</Type>
        <Flags>1</Flags>
      </Event>
      <RequiredField>
        <Name>PIN_FLD_POID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_EVENT</Name>
        <RequiredField>
          <Name>PIN_FLD_POID</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_START_T</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_END_T</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_BAL_IMPACTS</Name>
          <RequiredField>
            <Name>PIN_FLD_AMOUNT</Name>
```

```
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_RESOURCE_ID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_GL_ID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_IMPACT_TYPE</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_QUANTITY</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_RATE_TAG</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_TAX_CODE</Name>
      </RequiredField>
    </RequiredField>
  </RequiredField>
  <RequiredField>
    <Name>PIN_FLD_DISCOUNTS</Name>
    <RequiredField>
      <Name>PIN_FLD_ACCOUNT_OBJ</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_OWNER_OBJ</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_BAL_GRP_OBJ</Name>
    </RequiredField>
    <RequiredField>
      <Name>PIN_FLD_DISCOUNT_LIST</Name>
      <RequiredField>
        <Name>PIN_FLD_DISCOUNT_OBJ</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_PACKAGE_ID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_PURCHASE_END_T</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_PURCHASE_START_T</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_QUANTITY</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_STATUS</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_USAGE_END_T</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_USAGE_START_T</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_FLAGS</Name>
      </RequiredField>
      <RequiredField>
```

```
            <Name>PIN_FLD_TYPE</Name>
          </RequiredField>
        </RequiredField>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_BAL_INFO</Name>
        <RequiredField>
          <Name>PIN_FLD_BAL_GRP_OBJ</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_BALANCES</Name>
          <RequiredField>
            <Name>PIN_FLD_CURRENT_BAL</Name>
          </RequiredField>
        </RequiredField>
      </RequiredField>
    </EventMap>

<!--* Section which specifies fields sent when the event type is
    * any other value.-->

    <EventMap>
      <Event>
        <Type>*</Type>
        <Flags>1</Flags>
      </Event>
      <RequiredField>
        <Name>PIN_FLD_POID</Name>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_EVENT</Name>
        <RequiredField>
          <Name>PIN_FLD_POID</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_START_T</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_END_T</Name>
        </RequiredField>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_DISCOUNTS</Name>
        <RequiredField>
          <Name>PIN_FLD_ACCOUNT_OBJ</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_OWNER_OBJ</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_BAL_GRP_OBJ</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_DISCOUNT_LIST</Name>
          <RequiredField>
            <Name>PIN_FLD_DISCOUNT_OBJ</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_PACKAGE_ID</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_PURCHASE_END_T</Name>
```

```
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_PURCHASE_START_T</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_QUANTITY</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_STATUS</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_USAGE_END_T</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_USAGE_START_T</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_FLAGS</Name>
          </RequiredField>
          <RequiredField>
            <Name>PIN_FLD_TYPE</Name>
          </RequiredField>
        </RequiredField>
      </RequiredField>
      <RequiredField>
        <Name>PIN_FLD_BAL_INFO</Name>
        <RequiredField>
          <Name>PIN_FLD_BAL_GRP_OBJ</Name>
        </RequiredField>
        <RequiredField>
          <Name>PIN_FLD_BALANCES</Name>
          <RequiredField>
            <Name>PIN_FLD_CURRENT_BAL</Name>
          </RequiredField>
        </RequiredField>
      </RequiredField>
    </EventMap>
  </EventMapList>
</RTPTrimFlistConfiguration>
```

Measuring System Latencies with Instrumentation

You use the Pipeline Manager instrumentation feature to determine how much processing time each Pipeline Manager component (function modules, iScripts, and iRules) is consuming in microseconds. This information enables you to:

- Determine system benchmarks.
- Identify performance bottlenecks at the function module level.
- Add or reconfigure function pools to optimize CPU utilization.

Instrumentation collects statistics for the following components:

- The input module.
- Each function module.
- The output module.

After each transaction, the statistics for each pipeline tested are written to the **pipeline.log** file.

Using Instrumentation to Collect Module Performance Statistics

To enable instrumentation:

1. Start the pipeline.
2. Send a signal to the pipeline to toggle instrumentation on and off. Use the following commands to toggle the instrumentation state:

   Solaris/Linux:

   ```
   kill -s USR1 ifw_process_pid
   ```

   At the end of each transaction, the statistics are logged to the **pipeline.log** file and the statistics counters are reset.

   > **Note:**
   >
   > By default, Pipeline Manager instrumentation is disabled on startup. When Pipeline Manager is running, you can toggle between the disabled and enabled modes.

   > **Note:**
   >
   > Pipeline Manager begins gathering statistics immediately after receiving the signal. To assure accurate measurements, be sure that Pipeline Manager is not processing transactions when the signal is sent.

3. Process a sample CDR file.
4. Check the pipeline log files for processing time statistics. See "Viewing Instrumentation Testing Results" for more information.
5. When testing is complete, stop the instrumentation process by sending another signal. See step 2.

Viewing Instrumentation Testing Results

Each log file record consists of the fully qualified module name and the accumulated processing time spent in the module.

> **Note:**
>
> Pipeline processing time statistics are not cumulative. The output module writes data to a file whereas a function module processes EDRs in a different thread.

Sample Log File

The following sample log file shows instrumentation data:

```
15.03.2004 13:25:07 test         ifw          IFW       NORMAL   00516 -
 (ifw.Pipelines.ALL_RATE.Functions.PreProcessing) Plugin processing time statistics: 'ifw.Pipelines.ALL_R
ATE.Functions.PreProcessing.FunctionPool.APNMap
.Module, 7676390
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.CustomerRating.Module, 22629863
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.CustomerSearch.Module, 239523272
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.EventDiscarding.Module, 19874050
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.PreRatingZone.Module, 18751824
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.PreRecycle.Module, 1916139
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.PrefixDesc.Module, 8001348
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.ServiceCodeMap.Module, 4543899
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.UsageClassMap.Module, 6083775
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.UsageScenarioMap.Module, 9786078
ifw.Pipelines.ALL_RATE.Functions.PreProcessing.FunctionPool.UsageType.Module, 57114053
'
15.03.2004 13:25:07 test         ifw          IFW       NORMAL   00516 -
 (ifw.Pipelines.ALL_RATE.Functions.Rating) Plugin processing time statistics: 'ifw.Pipelines.ALL_RATE.Fun
ctions.Rating.FunctionPool.AddInfranetBillingRe
cord.Module, 44927730
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.MainRating.Module, 78250224
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.RateAdjustment.Module, 2358093
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.Recycle.Module, 1225628
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.Rejection.Module, 1785748
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.ServiceOutputSplit.Module, 6480466
ifw.Pipelines.ALL_RATE.Functions.Rating.FunctionPool.SpecialDayRate.Module, 8109825
'
15.03.2004 13:25:07 test         ifw          IFW       NORMAL   00516 -
 (ifw.Pipelines.ALL_RATE.Output.OutputCollection) Plugin processing time statistics: 'ifw.Pipelines.ALL_R
ATE.Output.OutputCollection.DevNull.Module, 67
ifw.Pipelines.ALL_RATE.Output.OutputCollection.DuplicateOutput.Module, 23
ifw.Pipelines.ALL_RATE.Output.OutputCollection.FAXOutput.Module, 561
ifw.Pipelines.ALL_RATE.Output.OutputCollection.GPRSOutput.Module, 728
ifw.Pipelines.ALL_RATE.Output.OutputCollection.RejectOutput.Module, 30
ifw.Pipelines.ALL_RATE.Output.OutputCollection.SMSOutput.Module, 552
ifw.Pipelines.ALL_RATE.Output.OutputCollection.TELOutput.Module, 178434585
ifw.Pipelines.ALL_RATE.Output.OutputCollection.WAPOutput.Module, 550
Note: To aggregate the counters into a nice report, please take a look at TracePerformanceReporting
```

Optimizing the DAT_USC_Map Module

The DAT_USC_Map module uses the Pipeline Manager framework component (FSM) to compile data mapping rules, which are stored in the database as regular expressions. The FSM compiles the data mapping structures only during Pipeline Manager startup because the rules can contain many comparisons of mapping patterns; this impacts startup performance. You can optimize the DAT_USC_Map module to enable Pipeline Manager to serialize the data structures and restore them from the serialized format.

About Precompiling USC Mapping Rules

Not all USC mapping data is stored in a compiled format: for example, rules used to define zone models. When the DAT_USC_Map module loads event data, it reorganizes it according to zone models to enable faster searching of the data structures during run time. This increases load time and memory requirements. To reduce the impact, you can configure Pipeline Manager to serialize the data structures the first time they are loaded and then reuse the serialized version during subsequent startup operations.

When Pipeline Manager begins processing data for a given zone model, it checks to see if a precompiled data file exists for that zone model. If so, it prepares the complex data structure by using the serialized format rather than by recompiling the structure from the USC map data.

If you enable the precompiling functionality, the following data is serialized:

- USC group
- Usage class and usage type
- Service code and service class
- Wholesale zone and retail zone

> **Note:**
>
> Data that is not in the precompiled format is read from the database or file system, depending on your DAT_USC_Map module configuration.

For more information, see "Precompiling Usage Scenario Mapping Data".

About Filtering Mapping Rules

You use USC groups to assemble the rules that define which services and service configurations are available to the pipeline; they contain the rules for mapping the service EDR attributes to each usage class.

You can configure your system to filter mapping rules based on USC groups so only the rules in the USC groups you specify are compiled and loaded into the DAT_USC_Map module. All other rules are ignored. This is more efficient than having one zone model that uses a large number of rules.

> **Note:**
>
> This is necessary only when your USC mapping rules are stored in the database; if they are read from a file, the data is already organized according to USC groups.

Generally you define USC Groups to contain the mapping rules for a specific type of EDR processing. For example, say you rate telephony services and process EDRs by using three USC groups (GSM, SMS, and GPRS), each of which contains mapping rules to determine domestic standard charges, domestic roaming charges, and international charges.

To increase performance, you can define the mapping rules for each set of charges in a separate zone model. Then, when an EDR is processed, based on the USC group specified, only the rules used in those zone models are compiled and loaded. This increases startup performance.

For more information, see "Filtering the Mapping Data to Compile and Load".

Configuring the DAT_USC_Map Module for Startup Performance

You improve startup performance of the DAT_USC_Map module by:

- Increasing the number of threads used to load mapping data.
- Precompiling usage scenario mapping data.
- Filtering the mapping data to compile and load.

You define these configurations in the Pipeline Manager registry file.

Increasing the Number of Threads Used to Load Mapping Data

The DAT_USC_Map module loads mapping rules for each zone model in a USC group by using a separate thread; therefore, it is only necessary to increase the number of threads when your USC groups contain multiple zone models.

To use multiple threads, set the **NumberOfThreads** registry entry to the desired number of threads. This enables Pipeline Manager to compile data in parallel and to restore it from the precompiled data files.

For example:

```
NumberOfThreads = 4
```

The default is **1**.

> **Note:**
>
> You can use this entry as a semaphore.

Precompiling Usage Scenario Mapping Data

To enable precompiling of USC mapping data, set the **PreCompiledDataDir** registry entry. This entry both enables the precompile functionality and defines the location of the compiled data files. By default, compiled data files are saved in the **./compiled_usc_data** directory.

Pipeline Manager saves them with the following naming convention:

USCzoneModelID.pc

For example, **GSM.pc**, **GSM_DOMESTIC.pc**, and **GSM_ROAMING.pc**.

If this entry is set, compiled files are created the next time the pipeline starts. For each subsequent run, the data files are validated against the data structures in the database and, if necessary, recompiled and resaved to the file system.

> **Note:**
>
> You can use this entry as a semaphore.

Filtering the Mapping Data to Compile and Load

If the source for your USC mapping rules is the database rather than a file, you can filter which rules are compiled and loaded into the DAT_USC_Map module when a pipeline starts by setting the **UscGroups**registry entry to one or more USC groups. For example:

```
UscGroups {GSM GSM_ROAMING}
```

> **Note:**
>
> You can specify only one USC group per each pipeline running in your system. If you use multiple USC groups, you must configure Pipeline Manager to run multiple pipeline instances. To do this, configure the Pipeline Manager registry file so the FCT_USC_Map in each pipeline instance refers to the appropriate DAT_USC_Map module reference and **UscGroups** entry.

By default, all mapping rules are loaded into the pipeline. see "About Filtering Mapping Rules" for more information.

> **Note:**
>
> You can use this entry as a semaphore.

Using Semaphores

You can use the new **NumberOfThreads**, **PreCompiledDataDir**, and **UscGroups** registry entries as semaphores to configure and control Pipeline Manager during pipeline startup. These semaphores perform the same tasks that the **Reload** semaphore performs, as specified in the startup registry or last-processed semaphore:

1. Load mapping data from the source (**Database** or **File**).
2. Create the USC zone model (from data in **PreCompiledDataDir** or **USCMapFile**).
3. Compile or precompile each USC zone model.

When you change the values of these semaphores after startup, they are not updated automatically in your system; you must use the **Reload** semaphore to update them during run time.

For example:

- To use multiple threads to load data, edit the **NumberOfThreads** semaphore and then call the **Reload** semaphore. Each thread processes a different zone model when loading the USC data.
- To reload USC data using a different set of files in the **PreCompiledDataDir** directory, edit the **PreCompiledDataDir** semaphore and then call the **Reload** semaphore.
- To filter a different set of mapping rules, edit the **UscGroups** semaphore and then call the **Reload** semaphore.

Other Pipeline Manager Monitoring Tools

This section describes additional Pipeline Manager performance-monitoring tools.

Viewing the Pipeline Log

You can see the results of tests for each pipeline in that pipeline's **pipeline.log** file.

> **Note:**
>
> Open each log file in a terminal windows and run **tail -f** on the logs.

After each batch stream is processed, the pipeline writes the following information to the **pipeline.log** files:

- The number of processed EDRs.
- The number of errors that occurred during EDR processing.
- The number of EDRs processed per second for a stream.
- If instrumentation is on, the instrumentation results. See "Viewing Instrumentation Testing Results" for more information.

Tuning Tips

- Let the system process a few files before you measure performance. This assures that any additional memory needed (for example, for the buffers) has been allocated.
- Use the system monitor tool to monitor system utilization.

Configuring Buffer Size Polling

Use the **QueueRequestTimeout** Controller entry in the registry to specify the interval in seconds that each queue's fill status is written to the log. For example:

```
ifw
{
  Active = TRUE
  ProcessLoopTimeout = 10
  QueueRequestTimeout = 10 # Optional, 0 disables
  ...
```

The default is **0** (no polling).

Buffer fill status information can indicate which function pool is the slowest. Over time, buffers in front of the slowest function pool fill up, and those that occur later in the stream are empty.

> **Note:**
>
> Instrumentation is the recommended tool for identifying the slowest function pool. See "Measuring System Latencies with Instrumentation".

OS-Specific Pipeline Manager Monitoring Tools

This section describes OS-specific tools that you can use to monitor and maintain your Pipeline Manager system.

Solaris Monitoring Tools

This section describes Solaris monitoring tools.

Displaying Thread Details

To display details of the threads within a process, use the **prstat** command:

```
prstat -Lmv -p
```

Example output:

```
prstat -Lmv -p 22376
   PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWPID
 22376 integ     86  13 0.0 0.0 0.0 0.0 0.0 0.9  12  3K .1M   0 ifw/4
 22376 integ     61  34 0.0 0.0 0.0 0.5 2.8 2.0 298  1K 64K   0 ifw/16
 22376 integ     52 0.8 0.0 0.0 0.0  42 0.0 4.9  56 11K 11K   0 ifw/117
 22376 integ     43 3.6 0.0 0.0 0.0  52 0.0 1.8 158  1K  7K   0 ifw/5
 22376 integ     22 0.1 0.0 0.0 0.0  75 0.0 2.6 393 125 463   0 ifw/116
 22376 integ     21 0.1 0.0 0.0 0.0  77 0.0 2.6  89 357 412   0 ifw/115
...
Total: 1 processes, 48 lwps, load averages: 4.18, 1.94, 2.38
```

In the pipeline **process.log** file, you can see when a thread is created, its name, and corresponding OS number:

```
09.12.2003 19:54:38 igscoll1     ifw          IFW       NORMAL   00000 -
(ifw.ProcessLog.Module) Thread instance ID '2'; and
Name 'ifw.ProcessLog.Module'.
...
09.12.2003 20:01:31 igscoll1     ifw          IFW       NORMAL   00000 -
(ifw.Pipelines.GSM.Input) Thread instance ID '16'; and
Name 'ifw.Pipelines.GSM.Input'.
...
09.12.2003 21:38:40 igscoll1     ifw          IFW       NORMAL   00000 -
(ifw.Pipelines.GSM.Functions.PreProcessing2) Thread instance ID '135'; and
Name 'ifw.Pipelines.GSM.Functions.PreProcessing2'.
...
```

Dumping the Call Stack of an Active Process

To dump the call stack of an active process, use the **vmstat** and **mpstat pstack** commands.

Identifying Thread Functions

To identify what each thread is used for, such as the pipeline framework, input, output, or function modules, use the **mpstat** command:

```
mpstat pstack
```

> **Note:**
>
> To determine the correlation between pipeline thread names and thread numbers, see the **pipeline.log** file.

Linux Monitoring Tools

Tools useful for monitoring Pipeline Manager on Linux systems include:

- **vmstat**
- **sar**
- **top**
- **pmap**
- **gnome-system-monitor**
- **sysstat package (iostat, mpstat, sadc, and sar)**

Configuring Multiple Pipelines in the DM to Improve Performance

By default, the front-end processes in the DMs write requests to a pipeline and the back-end processes listen to the pipeline and pick up the request. When all the back-end processes listen to a single pipeline, resources are wasted and performance might become slow. To improve performance, configure multiple pipelines in the DM, with each pipeline serving a set of back ends. The front-end processes send requests to the pipelines using the round-robin method. The back-end processes listen to the pipelines in order. For example, the first set of back-end processes listens to pipeline 1, the second set of processes listens to pipeline 2, and so on.

To configure multiple pipelines, include the following entry in the DM **pin.conf** file:

```
-dm dm_n_op_fifo nn
```

where *nn* is the number of pipelines. The number of your back-end processes, specified in the **dm_n_be** entry, must be a multiple of the pipelines you configure. The default is **1**.

> **Tip:**
>
> Start by configuring one pipeline for every six back-end processes and adjust the number of pipelines according to your requirements.

About Selective Account Loading

Pipeline Manager loads the subscriber data based on the service types configured for batch rating. If the service type is the same for both the prepaid and the postpaid subscribers, Pipeline Manager loads the prepaid subscriber data also.

You can configure your BRM system to load subscriber data selectively in Pipeline Manager based on the business profiles assigned to the accounts. For example, if you use selective account loading, you can load only data for postpaid services instead of postpaid and prepaid data, even though the service type is the same. You can configure any cache residency type data to be loaded into Pipeline Manager memory.

Selective account loading in Pipeline Manager provides:

- Reduced load time during initialization because less data is retrieved from the database.

- Improved memory usage because only selective subscriber information is stored in memory.

When rating the CDRs, Pipeline Manager treats the data as valid only if the cache residency value of the data at the time of the event matches with the values configured for loading data into Pipeline Manager.

Configuring Pipeline Manager for Selective Account Loading

You can configure Pipeline Manager to load selective accounts during initialization by enabling selective account loading functionality. See "Enabling Selective Account Loading".

> **Note:**
>
> For selective account loading functionality, you must enable the cache residency distinction parameter.

Enabling Selective Account Loading

By default, selective account loading functionality is disabled. You can enable this functionality by loading and configuring an optional business parameter, **CacheResidenciesForBatchPipeline**, in the *BRM_home***/sys/data/config/bus_params_selective_loading.xml** file.

To load and configure the values in the **CacheResidenciesForBatchPipeline** business parameter:

1. Search the **bus_params_selective_loading.xml** file for following line:

   ```
   <CacheResidenciesForBatchPipeline>0,1</CacheResidenciesForBatchPipeline>
   ```

2. Change **0,1** to any cache residency values of accounts you want to load, separated by comma. For example, to load convergent, prepaid, and postpaid accounts into Pipeline Manager, change **0,1** to **0,1,2**.
3. Save and close the file.
4. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_selective_loading.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. T
5. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
6. Stop and restart the CM.
7. Stop and restart Pipeline Manager.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter.

The following is a sample **bus_params_selective_loading.xml** file:

```
BusParamConfigurationClass>
  <BusParamsSelectiveLoading>
    <CacheResidenciesForBatchPipeline>0,1,2</      CacheResidenciesForBatchPipeline >
  </BusParamsSelectiveLoading>
</BusParamConfigurationClass>
```

Here, **0,1,2** specifies the cache residency types DEFAULT, REALTIME, and POSTPAID. After the **CacheResidenciesForBatchPipeline** business parameter is loaded, Pipeline Manager loads all accounts with Convergent, Prepaid, and Postpaid business profiles.

Configuring Pipeline Manager to Process Prepaid CDRs

By default, Pipeline Manager rates only one event type per service (for example, delayed session event for GSM telephony service).

If the selective account loading functionality is enabled, you can load prepaid subscribers in Pipeline Manager. However, Pipeline Manager rejects any prepaid CDRs of the prepaid subscribers if the delayed event type configured for batch rating is not present in any of the charge offers owned by the service or account. This is because of the difference in the prepaid and postpaid event types. For example, real-time session event for prepaid events and delayed session event for postpaid events.

To allow the CustomerSearch module to accept the prepaid CDRs, you can use the FCT_Account module **DisableRatingProductCheck** registry entry to configure how charge offer rating is checked:

- If you enable this entry, FCT_Account does not reject any prepaid CDRs of the prepaid subscribers if the configured event for batch rating is not present in any of the charge offers owned by the service or account. Pipeline Manager does not rate CDRs, but the DAT_AccountBatch plug-in provides the subscriber information. You can use this subscriber information for any customized processing. For example, to pass rated roaming prepaid CDRs through Pipeline Manager, you can customize the action on the CDRs based on the subscriber information.
- If you disable this entry, the FCT_Account rejects any prepaid CDRs of the prepaid subscribers if the configured event for batch rating is not present in any of the charge offers owned by the service or account. By default, **DisableRatingProductCheck** is set to **False**.

Running the purge_audit_tables.pl Script For Pipeline Manager

You can remove unwanted audit data from your Pipeline Manager audit tables. Purging the audit tables improves system performance, reduces memory usage, and makes the results returned by the DAT_Account module smaller and more efficient.

Use the **purge_audit_tables.pl** script to archive unneeded shadow objects in audit tables.

> **Note:**
>
> The **purge_audit_tables.pl** script does not delete objects from the database; it only purges the object rows stored in a table.

To purge objects from audit tables:

1. Open the *BRM_home***/sys/archive/oracle/purge_audit_tables.conf** file.
2. In the **storage_clause** entry, specify the tablespace for the history tables.
3. In the **time** entry, specify the column name to be used for comparing the cutoff date specified in the **purge_audit_tables.pl** script's **-d** parameter.

4. In the **cutoff_for_purge** entry, specify the percentage based on which it will invoke the **archiveindirect** mode rather than the **archivedirect** mode to archive the tables.

   For example, if the **cutoff_for_purge** value is **70**, and a table contains more then 70% data that must be archived, temporary tables are used to transfer the data efficiently (**archiveindirect** mode). If the table contains less then 70% data that must be archived, the data is transferred directly to the history tables (**archivedirect** mode).

   The *BRM_home***/sys/archive/oracle/purge_audit** file contains more information about the configuration entries.

5. With a text editor, open the *BRM_home***/sys/archive/oracle/purge_audit** file.
6. In the first line of the script, replace **__PERL__** with the location of the Perl executable.
7. Run the **purge_audit_tables.pl** script. See "purge_audit_tables.pl" in *BRM System Administrator's Guide* for more information.

> **Note:**
>
> To run in debug mode, set the environment variable ARCHIVE_DEBUG at the system prompt before you run the script. As the script runs, processing data, including the functions that are called, is printed to the screen.

Running the partition_utils Utility with Pipeline Manager

When you use Pipeline Manager batch rating, Rated Event (RE) Loader loads delayed events into the BRM database. You can enable partitioning for delayed events and create partitions for the delayed events.

Use the **partition_utils** utility to create database partitions. Before you use the **-f** parameter to force partitions, stop all BRM components, all instances of Pipeline Manager, and all instances of Rated Event Loader.

72

Migrating Accounts with the Pipeline Manager Running

This chapter explains:

- How BRM migrates accounts when Pipeline Manager is running.
- How to configure your Oracle Communications Billing and Revenue Management (BRM) system to migrate accounts when Pipeline Manager is running.

Before you read this chapter, you should be familiar with account migration and its configuration in BRM.

About Migrating Accounts When Pipeline Manager Is Online

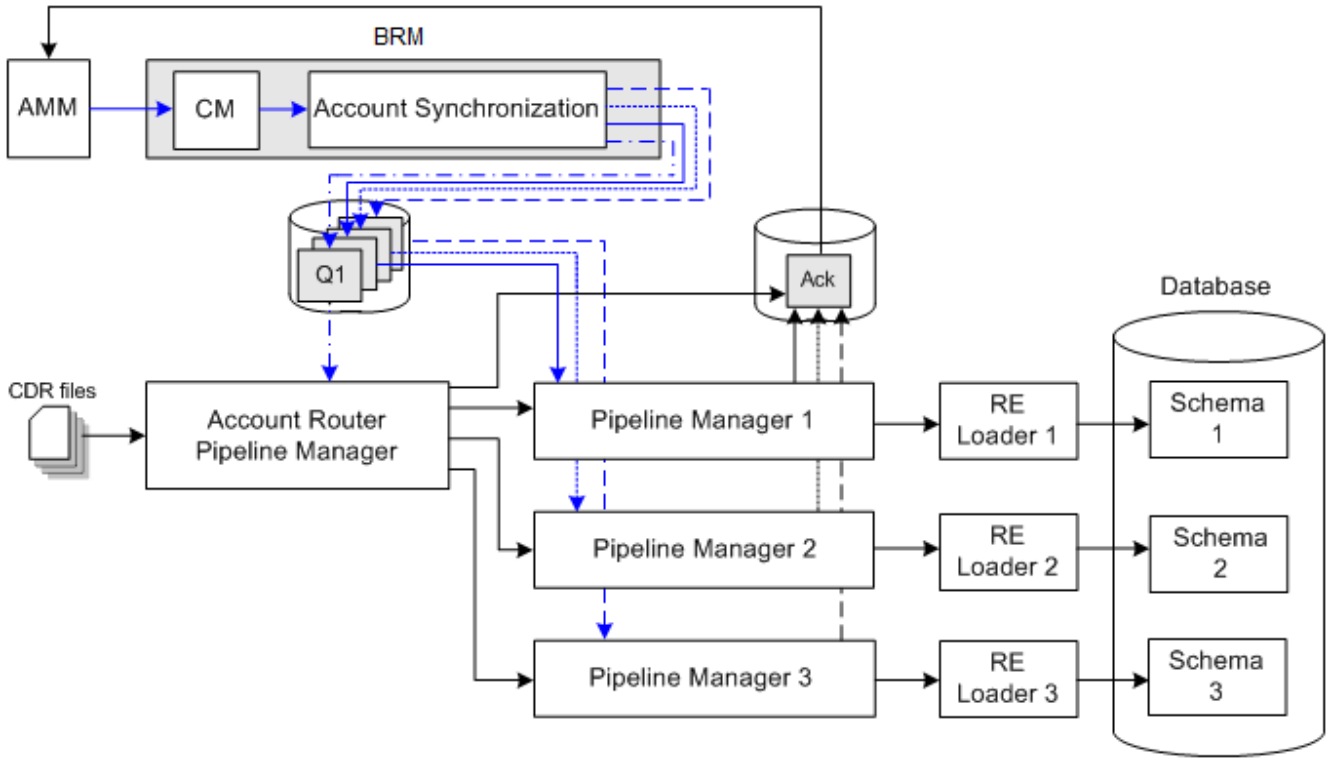
By default, AMM does not support migration when your pipelines are running. You specify whether AMM can migrate accounts while the Pipeline Manager is online by using the **controller**_*N*_**event_generation** parameter in the AMM **Infranet.properties** file.

BRM migrates accounts while the Pipeline Manager is running whether you use the AMM process (with its **pin_amt** utility) or the **pin_amt_tt** utility.

When you migrate accounts while Pipeline Manager is online, your pipelines stop processing any EDRs that apply to accounts undergoing migration. Your pipelines continue processing all other EDRs.

Figure 72-1 shows AMM interaction with the Pipeline Manager.

Figure 72-1 AMM Pipeline EDR Management



To coordinate account migration with your pipelines:

- AMM notifies the pipelines about a job's migration status by sending business events. See "About Notifying the Pipelines about Account Migration".
- The pipelines notify AMM about EDR processing status by sending acknowledgment events. See "About Notifying AMM about EDR Processing".
- The account-router Pipeline Manager suspends, recycles, and routes EDRs. See "About the Account Router Instance of the Pipeline Manager".

Do Not Rerate Events during Account Migration

Because the AMM software may suspend some events that you want to rerate, you *must not* rerate pipeline events during account migration.

How AMM Interacts with Your Pipelines during Account Migration

The following steps outline how AMM interacts with your pipelines when processing account migration jobs:

1. AMM fetches a configurable number of migration jobs. See "About Starting Multiple Jobs Concurrently".
2. AMM notifies the account-router Pipeline Manager to hold EDRs for all accounts in a job.
3. The account-router Pipeline Manager begins holding all EDRs for the specified list of accounts and sends an acknowledgment to AMM. See "About Suspending Call Records".

4. AMM waits a specified amount of time before migrating accounts. See "About Waiting before Migrating Accounts".
5. AMM migrates all accounts in the job.
6. AMM determines whether the job migrated successfully.
   - If migration finished successfully, AMM notifies the account router, source, and destination instances of the Pipeline Manager.

     > **Note:**
     >
     > If configured to do so, AMM also notifies any external applications.

   - If migration failed, AMM does not send any notification to your pipelines and job processing stops.

     > **Note:**
     >
     > When migration fails, your pipelines continue to suspend all EDRs for the specified accounts. You must fix the problem and remigrate the job before the pipeline can begin reprocessing suspended EDRs.

7. The account router, source, and destination instances of the Pipeline Manager update their account information and send an acknowledgment to AMM.
8. AMM notifies the account router to resume processing EDRs for the specified list of accounts.
9. The account router resumes processing EDRs for the specified accounts and sends an acknowledgment to AMM.
10. AMM calls the PCM_OP_SEARCH_RECYCLE opcode to recycle suspended EDRs through the pipeline. See "About Reprocessing Suspended Call Records".

About Waiting before Migrating Accounts

After the account-router Pipeline Manager begins suspending EDRs, AMM waits a configurable amount of time before migrating a job. This provides time for your pipelines to flush any EDRs targeted for accounts in the migration job.

The default wait time is 120 minutes. You specify how long the AMM Controller waits before migrating accounts by using the **Controller**_*N*_**hold_period** entry in the AMM **Infranet.properties** file.

About Starting Multiple Jobs Concurrently

You can minimize the amount of time AMM spends in the waiting period by configuring AMM to start multiple migration jobs concurrently. In this configuration, AMM:

1. Fetches a configurable number of jobs.
2. Notifies the account-router Pipeline Manager to hold EDRs for multiple jobs.
3. Starts the timer for each job.
4. Once the waiting period is over, AMM migrates jobs individually.

This increases the number of jobs in the queue that are ready to be migrated.

You specify how many jobs an AMM Controller processes concurrently by using the **Controller**_*N*_**concurrent_job_number** entry in the AMM **Infranet.properties file**.

About Notifying the Pipelines about Account Migration

AMM notifies your pipelines about account migration by sending a series of business events through the Account Synchronization architecture.

About AMM Business Events

AMM generates the five business events listed in Table 72-1 to notify the account router, source, and destination instances of the Pipeline Manager when account migration occurs:

Table 72-1 AMM Business Events

Event	Recipient	Description
HoldCDRProcessing	Account-Router Pipeline Manager	Notifies the account-router Pipeline Manager to suspend all EDRs for a specified list of accounts.
ResumeCDRProcessing	Account-Router Pipeline Manager	Notifies the account-router Pipeline Manager to resume processing all suspended and new EDRs for the specified list of accounts.
MigrateAcct	Account-Router Pipeline Manager External Applications	Notifies the account-router Pipeline Manager and any external applications to update the account database location for the specified list of accounts.
MigrateSource	Source Pipeline Manager	Notifies the source Pipeline Manager that all accounts in the job migrated successfully.
MigrateDestination	Destination Pipeline Manager	Notifies the destination Pipeline Manager that all accounts in the job migrated successfully. The destination pipeline then reads account information from the database.

About Sending AMM Business Events to the Pipelines

AMM sends business events to the pipelines by using a series of Account Synchronization queues. Each instance of the Pipeline Manager contains its own queue, which is dedicated to receiving business events from BRM and AMM.

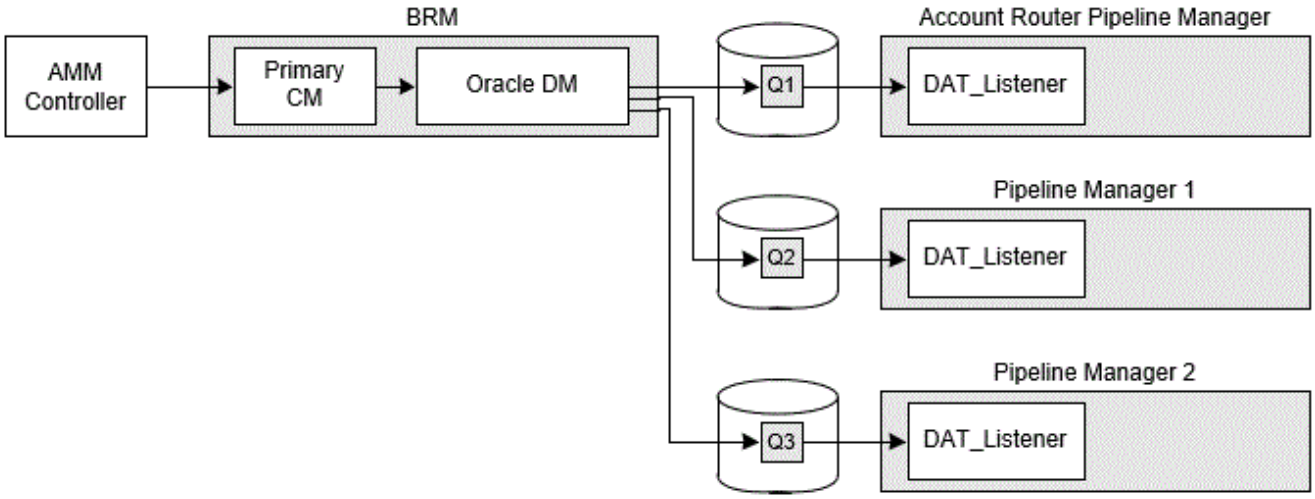
> **Note:**
>
> If configured to do so, AMM also sends business events to a queue for external applications.

AMM sends business events to a Pipeline Manager as follows:

1. AMM sends an event to the primary Connection Manager (CM).
2. The primary CM sends the event to the Account Synchronization architecture.
3. The Account Synchronization architecture uses its **ifw_sync_queuenames** file to publish the business event to the appropriate queue.

4. The Pipeline Manager's DAT_Listener module dequeues the event and then forwards it to the appropriate pipeline data module.

Figure 72-2 Sending AMM Business Events



You configure your system to send AMM business events to your pipelines by:

- Connecting AMM to the primary CM.
- Creating an Oracle database queue for each instance of the Pipeline Manager.
- Configuring Account Synchronization to publish AMM business events to your queues.
- Configuring each instance of the Pipeline Manager to dequeue AMM business events from its associated Account Synchronization queue.

Figure 72-2 shows the AMM business events process described. See "Configuring AMM to Send Business Events to Your Pipelines".

About Notifying AMM about EDR Processing

Your pipelines notify AMM when it begins holding EDRs, reprocessing EDRs, or updating account data by sending acknowledgment events through a dedicated acknowledgment queue.

About Acknowledgment Events

Each instance of the Pipeline Manager generates acknowledgment events when the following actions listed in Table 72-2 occur:

Table 72-2 Pipeline Manager Acknowledgment Events

Pipeline Instance	Sends Acknowledgments When...
Account router Pipeline Manager	• It begins suspending EDRs for a specified migration job. • It resumes processing EDRs for a specified migration job. • It completes an update of account locations in pipeline memory. This occurs after a migration job is successfully completed.

Table 72-2 (Cont.) Pipeline Manager Acknowledgment Events

Pipeline Instance	Sends Acknowledgments When...
Pipeline Managers connected to the BRM database	• It completes an update of account locations in pipeline memory. This occurs when accounts in a job are successfully migrated to its associated database schema. • It receives a **MigrateSource** event from AMM to indicate that accounts were successfully migrated away from its associated database schema.

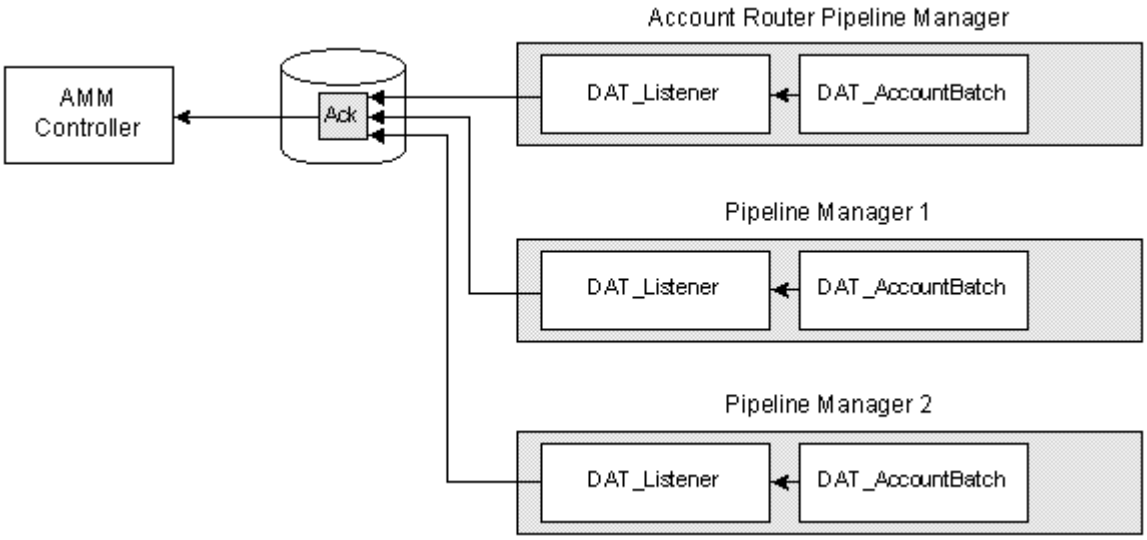
About Sending Acknowledgments to AMM

Each instance of the Pipeline Manager sends acknowledgment events to AMM by using a dedicated acknowledgment queue as shown in Figure 72-3.

Each Pipeline Manager sends acknowledgments as follows:

1. The DAT_AccountBatch module sends an acknowledgment event to the DAT_Listener module.
2. The DAT_Listener module publishes the event to the acknowledgment queue.
3. The AMM Controller dequeues the event.

Figure 72-3 Sending Pipeline Acknowledgment Events to AMM



To configure your pipelines to send acknowledgments to AMM, you must:

- Configure the DAT_Listener module in *each* instance of the Pipeline Manager to publish acknowledgment events.
- Create a single database queue that is dedicated to acknowledgment events. All instances of the Pipeline Manager use this single queue.
- Configure the AMM Controller to dequeue events from the acknowledgment queue.

For more information, see "Configuring Your Pipelines to Send Acknowledgments to AMM".

About the Account Router Instance of the Pipeline Manager

The account router instance of the Pipeline Manager is used in multischema systems to route EDRs to the correct instance of the Pipeline Manager. For example, EDRs targeted for accounts that reside in database schema 3 are routed to the Pipeline Manager instance associated with schema 3.

When configured for migration, the account-router Pipeline Manager also performs the following tasks:

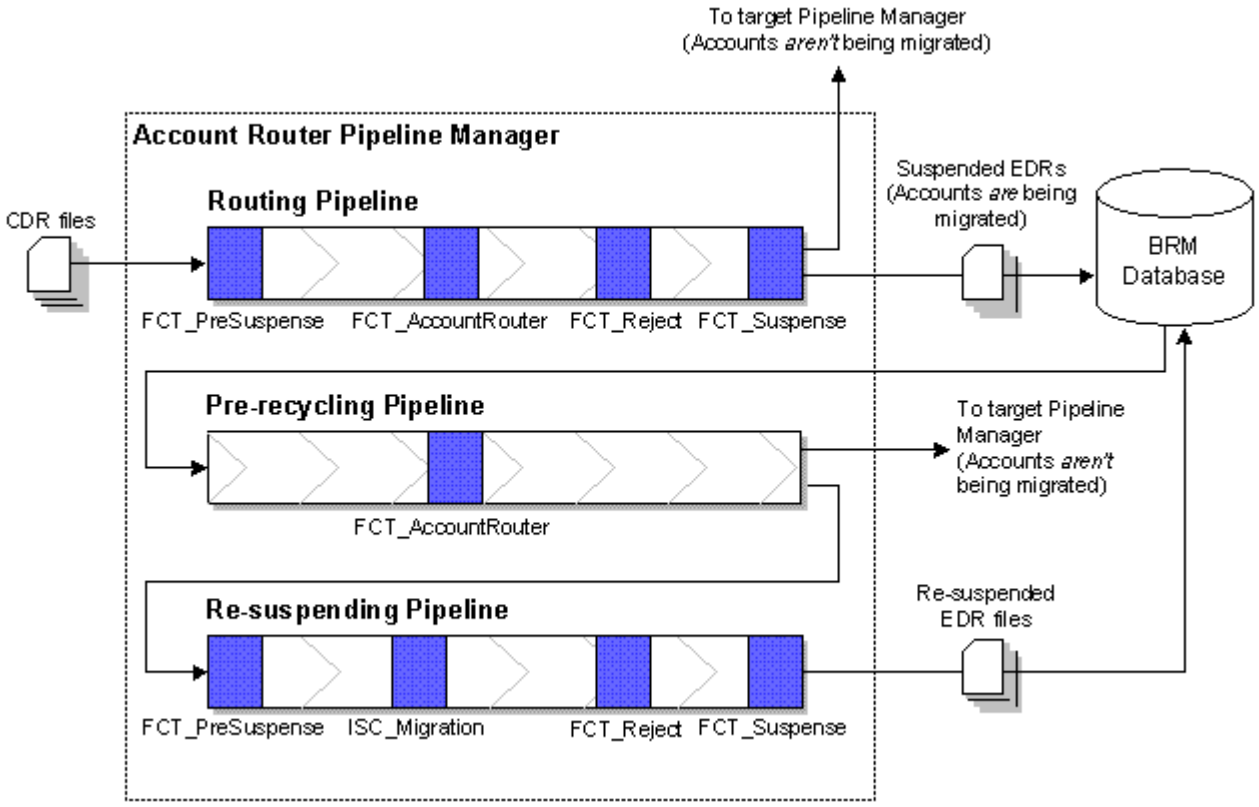
- Suspends EDRs targeted for accounts that are undergoing migration. See "About Suspending Call Records".
- Recycles previously suspended EDRs. See "About Reprocessing Suspended Call Records".

You configure the account-router Pipeline Manager for migration by creating three separate pipelines:

- A *routing pipeline* that routes EDRs to the appropriate instance of the Pipeline Manager and suspends any EDRs targeted for accounts undergoing migration. This pipeline must include the FCT_AccountRouter module, set to router mode; the FCT_PreSuspense module; the FCT_Reject module; and the FCT_Suspense module.
- A *pre-recycling pipeline* that processes previously suspended EDRs, determines whether an EDR is targeted for an account undergoing migration, and then routes the EDR to the appropriate output stream. This pipeline must include the FCT_AccountRouter module, set to recycle mode.
- A *resuspending pipeline* that automatically suspends all EDRs. This pipeline must include the FCT_PreSuspense module, the ISC_Migration iScript, the FCT_Reject module, and the FCT_Suspense module.

You must also configure the account router data pool to pass migration status information to your pipelines. Figure 72-4 shows the necessary pipelines and account router data pool.

Figure 72-4 Account Router Pipeline Manager



For information on how to configure the account-router Pipeline Manager, see "Configuring Your Account-Router Pipeline Manager".

About Suspending Call Records

The account-router Pipeline Manager initially routes and suspends call records in the *routing pipeline*.

After AMM notifies the account-router Pipeline Manager that a job is being migrated, the routing pipeline performs the following:

1. (Optional) The FCT_CallAssembly module assembles EDRs that were split into multiple records.

> **Note:**
>
> Any call assembling must occur in the account router instance of the Pipeline Manager and not in other instances.

2. (Optional) The FCT_DuplicateCheck module checks whether EDRs have been previously rated by the Pipeline Manager.

> **Note:**
>
> Any checking for duplicate EDRs must occur in the account router instance of the Pipeline Manager and not in other instances.

3. The FCT_PreSuspense module adds suspense-related data to the EDR.
4. The FCT_AccountRouter module, set to **Router** mode:
   - Flags the EDR for the target Pipeline Manager.
   - Determines whether an EDR is for an account undergoing migration. If it is, FCT_AccountRouter flags the EDR for suspension.
5. The FCT_Reject module routes EDRs with a specified error status, such as warning or critical, to the suspense output stream.
6. The FCT_Suspense module determines whether an EDR is flagged for suspension. If it is, FCT_Suspense places the EDR in a separate suspense output stream, where it is eventually loaded into the BRM database by the Suspense Event (SE) Loader.

> **Note:**
>
> You can use either standard recycling or Suspense Manager with AMM.

About Reprocessing Suspended Call Records

After AMM successfully migrates a job, it calls the PCM_OP_SEARCH_RECYCLE opcode to recycle previously suspended EDRs through the pipeline. Then, the account-router Pipeline Manager recycles suspended EDRs through the *pre-recycling pipeline* and the *resuspending pipeline*.

The account-router Pipeline Manager recycles EDRs as follows:

1. In the pre-recycling pipeline, the FCT_AccountRouter module, set to **Recycle** mode, determines whether an EDR is targeted for an account that is being migrated by a new job.
   - If the account *is* being migrated by a new job, FCT_AccountRouter flags the EDR for suspension and routes the EDR to a separate suspense output stream, where it is processed by the resuspending pipeline.
   - If the account *is not* being migrated, FCT_AccountRouter flags the EDR for the appropriate instance of the Pipeline Manager. The EDR is then rated by the target Pipeline Manager.
2. The resuspending pipeline automatically routes EDRs to a separate suspense output stream, which is eventually loaded into the BRM database by Suspense Event (SE) Loader.
   a. The FCT_PreSuspense module adds suspense-related data to the EDR.
   b. The ISC_Migration iScript automatically flags the EDR for suspension.
   c. The FCT_Reject module routes EDRs with a specified error status to the suspense output stream.
   d. The FCT_Suspense module routes the EDR to a suspense output stream, which is eventually loaded into the BRM database by SE Loader.

Configuring Your System to Migrate Accounts When the Pipeline Manager Is Running

You configure your BRM system to migrate accounts when your pipelines are online by:

1. Configuring Your Account-Router Pipeline Manager
2. Configuring BRM to Handle Suspended EDRs
3. Configuring AMM to Send Business Events to Your Pipelines
4. Configuring Your Pipelines to Send Acknowledgments to AMM

Configuring Your Account-Router Pipeline Manager

To configure your account router instance of the Pipeline Manager, perform the following:

- Configuring Your Routing Pipeline
- Configuring Your Pre-recycling Pipeline
- Configuring Your Resuspending Pipeline
- Configuring the Data Pool

Configuring Your Routing Pipeline

You configure your routing pipeline to route and suspend EDRs by using the following pipeline modules:

- **FCT_PreSuspense**. To make suspense fields queryable in Suspense Management Center, set the following FCT_PreSuspense registry entries:
  - Use the **Active** entry to enable this module.
  - Use the **QueryableFields** entry to specify the tables and fields that you can perform queries on in Suspense Management Center.
- **FCT_AccountRouter** set to **Router** mode. To flag EDRs for suspension and for the appropriate Pipeline Manager, set the following FCT_AccountRouter registry entries:
  - Use the **Active** entry to enable this module.
  - Use the **Mode** entry to specify **Router** mode.
  - Use the **Streams** entry to map EDRs to the appropriate output stream.
- **FCT_Reject**. To route EDRs with a specified error status to the suspense output stream:
  - Use the **Active** entry to enable this module.
  - Set the **UseRejectStream** entry to **True**. This sends EDRs to the reject stream.
  - Use the **MinErrorSeverity** entry to reject EDRs that have the specified error severity.
  - Use the **StreamMap** entry to map errors to specific output streams.

> **Note:**
>
> You must also configure an instance of the Out_Reject module for rejected EDRs. All rejected EDRs must be set to the suspense output stream.

- **FCT_Suspense**. To send EDRs to the suspense output stream, set the following FCT_Suspense registry entries:
  - Use the **Active** entry to enable this module.
  - Use the **SuspenseCreateStream** entry to specify the output stream for suspended EDRs.
  - Use the **SuspenseUpdateStream** entry to specify the output stream for recycled EDRs.
  - Use the **DataConnection** entry to specify how to connect to the BRM database.

If you want your pipelines to assemble EDRs or check for duplicate EDRs, you must also use the FCT_CallAssembly and FCT_DuplicateCheck modules in the routing pipeline.

Configuring Your Pre-recycling Pipeline

You configure your pre-recycling pipeline to recycle or suspend EDRs by using the FCT_AccountRouter module set to **Recycle** mode. Make sure you also set the following FCT_AccountRouter registry entries:

- Use the **Active** entry to enable this module.
- Use the **Mode** entry to specify **Recycle** mode.
- Use the **Streams** entry to map EDRs to the appropriate output stream.

Configuring Your Resuspending Pipeline

You configure your resuspending pipeline to automatically suspend all EDRs by using the following pipeline modules:

- **FCT_PreSuspense**. To make suspense fields queryable in Suspense Management Center, set the following FCT_PreSuspense registry entries:
  - Use the **Active** entry to enable this module.
  - Use the **QueryableFields** entry to specify the tables and fields that you can perform queries on in Suspense Management Center.
- **ISC_Migration**. To automatically flag all EDRs for suspension, set the following ISC_Migration registry entries:
  - Use the **Active** entry to enable this module.
  - Use the **Filename** entry to specify the path to the ISC_Migration file.
- **FCT_Reject**. To route EDRs with a specified error status to the suspense output stream:
  - Use the **Active** entry to enable this module.
  - Set the **UseRejectStream** entry to **True**. This sends EDRs to the reject stream.
  - Use the **MinErrorSeverity** entry to reject EDRs that have the specified error severity.
  - Use the **StreamMap** entry to map errors to specific output streams.

> **Note:**
>
> You must also configure an instance of the Out_Reject module for rejected EDRs. All rejected EDRs must be set to the suspense output stream.

- **FCT_Suspense**. To send EDRs to the suspense output stream, set the following FCT_Suspense registry entries:
  - Use the **Active** entry to enable this module.
  - Use the **SuspenseCreateStream** entry to specify the output stream for suspended EDRs.
  - Use the **SuspenseUpdateStream** entry to specify the output stream for recycled EDRs.
  - Use the **DataConnection** entry to specify how to connect to the BRM database.

Configuring the Data Pool

You configure the account-router Pipeline Manager data pool to pass account migration data to your pipelines by using the following pipeline data modules:

- **DAT_AccountBatch** stores AMM business events. In addition to setting the standard connection registry entries:
  - Set the **UseAsRouter** entry to **True**. This is required.
  - (Optional) Use the **PrintAMTData** entry to specify whether to print AMM data to a log file. You can use this data for troubleshooting.
  - (Optional) Use the **PrintAMTJobData** entry to specify whether to print data about one migration job to a log file. You can use this data for troubleshooting.
- **DAT_BalanceBatch.** To provide accurate account balances during migration, set the following DAT_BalanceBatch registry entries:
  - Use the **IntegrateConnection** entry to specify how to connect to the pipeline database. This entry points to the **Login** registry section.
  - Use the **InfranetConnection** entry to specify how to connect to the BRM database. This entry points to the **LoginInfranet** registry section.
  - Use the **ListenerDataModule** entry to specify how to connect to the DAT_Listener module. This entry points to the **Listener** registry section.
- **DAT_Listener.** To retrieve business events from BRM and send acknowledgment events directly to the acknowledgment queue, set the following DAT_Listener registry entries:
  - Use the **InfranetConnection** entry to specify how to connect to the database schema that contains your queues. This entry points to the **LoginInfranet** registry section.
  - Use the **AckQueueNameAMM** entry to specify the name of the acknowledgment queue.
  - Use the **QueueName** entry to specify the name of the Account Synchronization queue that stores the AMM business events.

Configuring BRM to Handle Suspended EDRs

BRM offers both the default *standard recycling* feature and the optional *Suspense Manager* feature to recycle EDRs.

AMM works with both standard recycling and Suspense Manager.

Both standard recycling and Suspense Manager enable you to do the following:

- Load suspended EDRs into the BRM database.
- View, edit, write off, or recycle suspended EDRs.
- Retrieve suspended EDRs from the BRM database.

You must configure your pipeline before you can migrate accounts with the pipeline running. If you have purchased Suspense Manager, see the Suspense Manager documentation for further configuration instructions.

Configuring AMM to Send Business Events to Your Pipelines

To configure AMM to send business events to your pipelines, perform the following:

1. Connecting AMM to the Primary CM
2. Configuring Account Synchronization
3. Configuring Your Pipelines to Dequeue AMM Business Events

Connecting AMM to the Primary CM

You connect AMM to the primary Connection Manager (CM) so that AMM can send business events to the Account Synchronization architecture, where they are eventually routed to your pipelines.

You connect AMM to the primary CM by using the **infranet.connection** and **infranet.login.type** parameters in the AMM **Infranet.properties** file.

Configuring Account Synchronization

You configure the Account Synchronization architecture to send AMM business events to the appropriate instance of the Pipeline Manager by performing the following:

1. Configuring Account Synchronization to send BRM events to your pipelines.
2. Creating queues for sending AMM business events to your pipelines. You must create a queue for each Pipeline Manager instance in your system. See "Configuring Your Account Synchronization Queues for AMM Business Events".
3. Mapping the AMM business events to your Oracle database queues. See "Mapping AMM business events to your queues".

Configuring Your Account Synchronization Queues for AMM Business Events

The Account Synchronization framework uses a set of Oracle database queues to send both BRM events and AMM business events to your pipelines.

Each instance of the Pipeline Manager must have its own database queue. For example, if your system contains three BRM database schemas, Account Synchronization requires a total of four queues. That is, one for each of the following instances:

- Account-router Pipeline Manager
- Pipeline Manager for BRM database schema 1
- Pipeline Manager for BRM database schema 2
- Pipeline Manager for BRM database schema 3

> **Note:**
>
> You must also create a separate queue for any external applications that require notifications about account migration.

If your system does not already contain a queue for each instance of the Pipeline Manager, you can create additional queues by using the Account Synchronization **pin_ifw_sync_oracle** utility.

Mapping AMM business events to your queues

You map which types of events Account Synchronization sends to your queues by using the **ifw_sync_queuenames** file (*BRM_home*/**sys/dm_ifw_sync/ifw_sync_queuenames**). This file lists all queues in your system and the events to route to each one.

You configure the **ifw_sync_queuenames** file to map AMM business events, in addition to your BRM events, to each of your queues by using the following syntax:

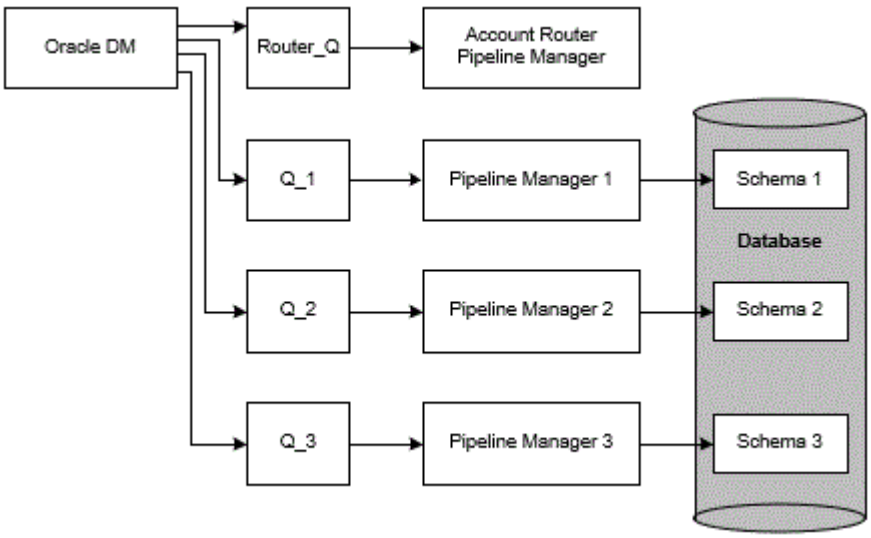
```
QueueName@DatabaseLink
{
    Criteria
}
```

Where:

- *QueueName* specifies the name of the queue.
- *DatabaseLink* specifies the database link for connecting to queues on other database schemas. Provide a link only for queues that reside on a separate schema from which the Account Synchronization DM connects.
- *Criteria* specifies which events to send to the queue. You can configure the Account Synchronization DM to send all business events, only events from a specific database schema, or only specific event types.

For example, assume your BRM system contains three database schemas and four queues as shown in Figure 72-5:

Figure 72-5 Sample Pipeline Account Synchronization Architecture



In this system, you must configure the **ifw_sync_queuenames** file so that the Account Synchronization architecture does the following:

- Sends **HoldCDRProcessing**, **ResumeCDRProcessing**, and **MigrateAcct** events to the queue for the account-router Pipeline Manager
- Sends **MigrateSource** and **MigrateDestination** events to each queue connected to a BRM database schema

For more information about AMM business events, see "About Notifying the Pipelines about Account Migration".

In this example, the **ifw_sync_queuenames** file requires the following additional entries:

```
ROUTER_Q
{
    HoldCDRProcessing
    ResumeCDRProcessing
    MigrateAcct
}

Q_1 # local database schema queue
{
    MigrateSource
    MigrateDestination
}
Q_2@database_link_1 # remote database schema
{
    MigrateSource
    MigrateDestination
}
Q_3@database_link_2 # remote database schema
{
    MigrateSource
    MigrateDestination
}
```

Configuring Your Pipelines to Dequeue AMM Business Events

You must configure *each instance* of the Pipeline Manager to retrieve AMM business events from the Account Synchronization queue. To do this, connect the DAT_Listener modules to your Account Synchronization queues by setting the following registry entries:

- Use the **InfranetConnection** entry to specify how to connect to the database schema that contains the Account Synchronization queue. This entry points to the **LoginInfranet** registry section.
- Use the **QueueName** entry to specify the name of the Account Synchronization queue that holds the AMM business events.

Configuring Your Pipelines to Send Acknowledgments to AMM

You configure your pipelines to send acknowledgment events to a centralized queue, where they are retrieved by the AMM Controller, by performing the following:

1. Creating the Acknowledgment Queue
2. Connecting AMM Directly to Your Acknowledgment Queue
3. Configuring Your Pipelines to Send Acknowledgment Events

Creating the Acknowledgment Queue

You create a centralized acknowledgment queue for sending events from your pipelines to the AMM Controller.

You create the acknowledgment queue by using the **pin_ifw_sync_oracle** utility. Enter the following commands at a UNIX prompt:

```
% su - pin
% cd BRM_home/apps/pin_ifw_sync
% pin_ifw_sync_oracle.pl create [-l /@DatabaseAlias] [-q queue_name -t queue_table]
```

The utility creates a database queue named IFW_SYNC_QUEUE and a queue table named IFW_SYNC on the specified schema. To use nondefault names, use the **-q** and **-t** options to specify names for the queue and queue table.

> **Note:**
>
> In multischema systems, all queues and queue tables must use unique names. You must also make sure the acknowledgment queue is accessible by the **pin** user.

Connecting AMM Directly to Your Acknowledgment Queue

You connect the AMM Controller to the acknowledgment queue so that it can retrieve acknowledgment events.

You connect the AMM Controller to the acknowledgment queue by using the **controller**_*N*_**amt_queue_name** and **controller**_*N*_**amt_queue_owner_name** entries in the AMM **Infranet.properties** file.

Configuring Your Pipelines to Send Acknowledgment Events

You configure your pipelines to send acknowledgment events to AMM by configuring the DAT_Listener module in each Pipeline Manager.

Configure the DAT_Listener registry entries in *each* instance of the Pipeline Manager to specify the following:

- Use the **InfranetConnection** entry to specify how to connect to the database schema that contains your acknowledgment queue.
- Use the **AckQueueNameAMM** entry to specify the name of the acknowledgment queue. This is the queue you created in "Creating the Acknowledgment Queue".

73
Pipeline Manager Error Messages

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager error messages.

> **Note:**
>
> - Many error descriptions include the string *value*. This string is replaced with the appropriate value by the module logging the error.
> - Modules that are not listed in this chapter do not log module-specific error messages. However, modules can return pipeline framework error messages. For information on framework error messages, see "Pipeline Framework Error Messages".

Pipeline Framework Error Messages

Table 73-1 lists the Pipeline Framework error messages.

Table 73-1 Pipeline Framework Error Messages

Error Message	Description
ERR_A_CUSTOMER_NOT_FOUND	A-Customer not found (*value*).
ERR_ACTIVATED_DATE_INVALID	Contract *value* has an invalid activation date.
ERR_ADD_DATABLOCK	Cannot add Datablock '*value*' to EDR-C.
ERR_ALIAS_IS_IN_WRONG_BLOCK	The alias *value* is in the wrong block.
ERR_BAD_SCHEMA_SOURCE	Bad schema source: *value*.
ERR_BLOCK_DESC_NOT_FOUND	Block description not found (*value*).
ERR_BLOCKID_UNKNOWN	Can't find an index for id *value*.
ERR_BUILD_DESC_TREE	Error while opening/reading description tree (*value*).
ERR_CALENDAR_PLUGIN_INV	Calendar data module invalid.
ERR_CALLTYPE_INVALID	Invalid call type: *value*: *value*.
ERR_CALLTYPE_NOT_FOUND	No call type found for EDR (*value*).
ERR_CAN_NOT_GET_FACTORY	Cannot get factory '*value*'.
ERR_CANCEL_FAILED_INPUT	Cancel failed in input-controller.
ERR_CANCEL_TRANSACTION	Module '*value*' failed to cancel transaction '*value*'.
ERR_CANNOT_DECRYPT_PASSWORD	Cannot decrypt password '*value*'; user '*value*'.
ERR_CANNOT_FIND_EVENT_HANDLER_PROC	Cannot locate the event handler daemon!
ERR_CANNOT_FORK	Cannot create child process.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_CANNOT_GET_DMT_DMCONNECTION	*value*: Cannot get a DMT::DMConnection
ERR_CANNOT_INIT_DB_VERSION	Cannot initialize database version, error '*value*'.
ERR_CANNOT_INIT_INPUT_STREAM	Cannot initialize the input stream object.
ERR_CANNOT_INIT_INPUT_STREAM_INTERFACE	Cannot initialize the input stream interface object.
ERR_CANNOT_INIT_OUTPUT_COLLECTION	Cannot initialize the output collection.
ERR_CANNOT_INIT_OUTPUT_MODULE	Cannot initialize the output module.
ERR_CANNOT_INIT_OUTPUT_STREAM	Cannot initialize the output stream object.
ERR_CANNOT_INIT_OUTPUT_STREAM_INTERFACE	Cannot initialize the output stream interface object.
ERR_CANNOT_JOIN_EVENT_HANDLER_PROC	Cannot connect to event handler process: *value*
ERR_CANNOT_OPEN_DATABASE	Cannot open database '*value*'; user '*value*'; password '*value*'; server message '*value*'.
ERR_CANNOT_RENAME_OUTPUT_FILE	Cannot rename temporary output file '*value*'.
ERR_CHARGE_ITEM_INVALID	ChargeItem *value* invalid.
ERR_CHARGED_ZONE_NOT_FOUND	The EDR charged zone cannot be found for '*value*'.
ERR_CIBER_RET	CIBER return: retReason *value*, retCode *value*, fieldID *value*, ruleID *value*.
ERR_CLIMAP_FILENAME_EMPTY	Empty **cli** mapping file name specified.
ERR_COMMIT_TRANSACTION	Module '*value*' failed to commit transaction '*value*'.
ERR_CON_ATTACHED_FIELD	The attached field information has not the right format. Must be BLOCKNAME.FIELDNAME, is *value*).
ERR_CONTROLLER_CONFIGURATION	Pipeline controller configuration has error in '*value*'.
ERR_CONTROLLER_HAS_WRONG_TYPE	Pipeline controller has wrong type in '*value*'.
ERR_CONVERSION_BAS_DATE	*value* could not be converted to BAS_Date.
ERR_CONVERSION_FAILED	EDR conversion failed (*value*).
ERR_CONVERSION_INT	*value* is no valid integer value.
ERR_CORBA_EXCEPTION	CORBA exception: *value*.
ERR_CREATE_ALIAS_MAP_INDEX	No AliasMap entry found for Reference '*value*' and logical Name '*value*'.
ERR_CREATE_EDR_INDEX	EDR index creation failed: *value* (name=`*value*', key=`*value*' and reference=`*value*').
ERR_CREATE_INDEX	EDR index creation failed: *value*
ERR_CREATE_INPUT_PARSER	Failed to create input parser: *value*
ERR_CREATE_INSTANCE	Error creating instance of *value*
ERR_CREATE_OBJECT_FAILED	Cannot create object '*value*' (invalid (NULL) pointer).
ERR_CREATE_OUTPUT_PARSER	Failed to create output parser: *value*
ERR_CREATE_SCRIPT	Error loading script *value*: *value*.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_CREATE_THREAD_FAILED	Cannot create thread instance for '*value*'; invalid thread body.
ERR_CUG_FILENAME_EMPTY	Empty closed user group file name specified.
ERR_CUST_A_IDENTIFICATION_UNKNOWN	Customer identification technique for used service ('*value*') not found
ERR_CUST_A_VALUE_NOT_FOUND	Missing value for field '*value*' of Customer A
ERR_CUST_FILE_VERSION	Illegal customer file version *value*.
ERR_CUST_FILENAME_EMPTY	Empty customer file name specified.
ERR_CUSTOMER_DATA_INVALID	Invalid customer data.
ERR_DAT_PREFDESC_INS_TREE_DB	Can't insert line *value* from table *value* into prefix description table.
ERR_DAT_PREFDESC_INS_TREE_FILE	Can't insert line *value* from file *value* into prefix description table.
ERR_DATA_INVALID	The data in field *value* in invalid.
ERR_DATA_PLUGIN_INVALID	Module '*value*' is invalid.
ERR_DATA_PLUGIN_NOT_FOUND	Module '*value*' cannot be found in the DataPool.
ERR_DATABASE	Database error '*value*'.
ERR_DB_COMMIT_TRANSACTION	Cannot commit database transaction '*value*'.
ERR_DB_CONNECTION_MODULE	Database connection module is invalid.
ERR_DB_CONNECTION_NOT_VALID	Could not connect to database.
ERR_DB_NUMBER_OF_ROWS	Statement '*value*' does not return exactly one row.
ERR_DB_START_TRANSACTION	Error starting database transaction: '*value*'
ERR_DB_STATEMENT_EXECUTE	Cannot execute database statement '*value*', message '*value*'.
ERR_DB_VERSION_CHECK	Wrong database version. Check module and database version.
ERR_DB_VERSIONS_NOT_FOUND	Database versions '*value*' not found.
ERR_DD_NOT_READ	Cannot read the data dictionary.
ERR_DEF_IS_INCOMPLETE	The field definition '*value*' is incomplete.
ERR_DEFAULT_BLOCK_NOT_FOUND	The specified default block name *value* does not exist in the description.
ERR_DEFAULT_WITH_WRONG_ID	Output Stream *value* : The default block has a wrong id. Check your format description.
ERR_DELETE_FILE	Cannot delete file '*value*'.
ERR_DELETE_OUTPUT_FILE	'*value*': Cannot delete output file '*value*'.
ERR_DIR_EMPTY	Reading from empty directory '*value*'.
ERR_DIR_NOT_ACCESSIBLE	Directory '*value*' is not accessible.
ERR_DIR_NOT_WRITEABLE	Directory *value* is not writable.
ERR_DLOPEN_FAILED	Cannot open shared library '*value*'; *value*. Make sure the LD_LIBRARY_PATH_64 environment variable includes *pipeline_home*/**lib**.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_DLSYM_FAILED	Cannot get address of generator function '*value*'; *value*.
ERR_DONE_PATH_NOT_FOUND	Entry for done path not found in registry.
ERR_DOUBLE_ALIAS_NAME	The reference to the alias *value* exist more than one times.
ERR_DOUBLE_SEQ_NUMBER	Double sequence number found (sequence number: '*value*').
ERR_DOUBLE_TRANS_MODULE	Transaction module '*value*' was attached more than once.
ERR_DOUBLE_TRANSACTION_ID	Double transaction id '*value*' found.
ERR_DUPLICATE_IRULE_PARAMETER	Duplicate iRule parameter '*value*' found in file '*value*'.
ERR_DUPLICATE_NUMPORTDATA	*value:* Duplicate number portability data found for the CLI *value* and the Portation TimeStamp *value*
ERR_EDR_ALIAS_NOT_FOUND	The specified field alias *value* couldn't founded.
ERR_EDR_BLOCK_NOT_FOUND	The specified block alias *value* couldn't founded.
ERR_EDR_BUILD_RECORD_NOT_FILLED	'*value*' - EDR buildt record field not filled.
ERR_EDR_CREATE	Failed to create new EDR container.
ERR_EDR_FACTORY_NOT_FOUND	EDR-Factory '*value*' not found.
ERR_EDRTRACE_STREAMLOG_CREATION_FAIL	Error in EDR trace stream log creation.
ERR_EMPTY_CHARGEPACKET_LIST	No charge-packets found in charge breakdown record.
ERR_ERROR_PATH_NOT_FOUND	Entry for error path not found in registry.
ERR_ERROR_RATE_ALREADY_DEFINED	Error rate for '*value*' already specified.
ERR_ERROR_RATE_VALUE_NOT_SPECIFIED	No value specified for error '*value*'.
ERR_EVAL_ENVIRONMENT	Cannot evaluate environment '*value*'.
ERR_FAILURE_FSM	Failure in finite state machine: *value*.
ERR_FILE_CLOSE_OS	*value*: Cannot close file '*value*'.
ERR_FILE_EOF	Tried to read past end of file '*value*'.
ERR_FILE_EXIST	File '*value*' exist.
ERR_FILE_MOVE_OS	*value*: Cannot move file '*value*' to '*value*'.
ERR_FILE_NOT_FOUND	File '*value*' not found.
ERR_FILE_NOT_MOVED	File '*value*' could not be moved to '*value*'.
ERR_FILE_NOT_WRITABLE	File '*value*' is not writable.
ERR_FILE_OPEN_OS	*value*: Cannot open file '*value*'.
ERR_FILE_READ_ERR	Error reading from file '*value*'.
ERR_FILE_READ_OS	*value*: Error reading from file '*value*'.
ERR_FILE_REMOVE_OS	*value*: Cannot remove file '*value*'.
ERR_FILE_WRITE_ERR	Error writing into file '*value*'.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_FILE_WRITE_OS	*value*: Error writing into file '*value*'.
ERR_FILENAME_MISSING	File name not set for '*value*'.
ERR_FLIST_INPUT_ERROR	Error while processing FLIST message: *value*
ERR_GAP_IN_SEQ_NUMBER	Gap in sequence number found (sequence number: '*value*').
ERR_GETTING_DATADESCR	Failed to get the data description.
ERR_GRAMMAR_SYMBOL_LOOKUP	Symbol lookup for `*value*' failed: *value*
ERR_ILL_RECORD_TYPE	Illegal record type '*value*' found.
ERR_ILLEGAL_STREAM_NUM	Tried to use illegal stream number '*value*' for output.
ERR_IN_RECEIVED_MESSAGE	Message *value* was invalid.
ERR_IN_SECTION	Error in section *value*.
ERR_INCORRECT_FILLER_LENGTH	Invalid record Filler length, expected: '*value*', received: '*value*'.
ERR_INCORRECT_FORMAT_OBJ	The format description object couldn't be founded or is invalid.
ERR_INDEX_NOT_CREATED	Couldn't create the index for alias *value*.
ERR_INDEX_NOT_FOUND	Container index not found.
ERR_INIT_EDR_ITERATOR	Failed to initialize EDR iterator for `*value*'.
ERR_INIT_SEG_TARIFF_LINK	Failure during initialization of tariff segment link table.
ERR_INIT_TSC_MAPTABLE	Failed to init map table: *value*.
ERR_INIT_XERCES	Error: Xerces-c Initialization. Exception message: *value*
ERR_INPUT_DONE_FILE_NOT_MOVED_TO_ERR	'*value*': Cannot move done file '*value*' to error file '*value*'.
ERR_INPUT_DONE_FILE_NOT_MOVED_TO_INPUT	'*value*': Cannot move done file '*value*' to input file '*value*'.
ERR_INPUT_FILE_NOT_MOVED	'*value*': Cannot move input file '*value*' to temporary file '*value*'.
ERR_INPUT_MAPPING_FAILED	Input mapping `*value*' failed: *value*.
ERR_INPUT_PATH_NOT_FOUND	Entry for input path not found in registry.
ERR_INPUT_REQUEST_ROLLBACK	The input has requested a rollback (reason=*value*).
ERR_INPUT_TEMP_FILE_NOT_MOVED	'*value*': Cannot move temporary input file '*value*' to input file '*value*'.
ERR_INPUT_TEMP_FILE_NOT_MOVED_TO_DONE_ERR	'*value*': Cannot move temporary file '*value*' to done or err file '*value*'.
ERR_INSERT_HASH	Failure during insert in hash map.
ERR_INSERTING_CLI	Error loading **cli** '*value*' (probably duplicated)
ERR_INSUFFICIENT_MEMORY	Insufficient memory available.
ERR_INVALID_DATABASE_VALUE	Database value for field '*value*' is invalid.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_INVALID_DATE	Can't build date '*value*' for **cli** '*value*'.
ERR_INVALID_DATETIME	*value*. Cannot build **datetime** '*value*' for **cli** '*value*'.
ERR_INVALID_FCI_COLL_ENTRIES	Invalid number of FCI collection entries (*value*).
ERR_INVALID_FCI_COLL_ORDER	Invalid order of FCI collection entries (*value*). This error occurs when buffers are configured in multiple function pools. In this configuration, each buffer must have a unique name.
ERR_INVALID_FIRST_CALL_TIMESTAMP	Invalid first call timestamp: *value*, calculated: *value*.
ERR_INVALID_HA_ROLE	The peer instance has already assumed the *value* role
ERR_INVALID_INPUT_RECORD	Check length, numeric values or date fields for their correctness. (record:*value*)
ERR_INVALID_LAST_CALL_TIMESTAMP	Invalid last call timestamp: *value*, calculated: *value*.
ERR_INVALID_LINE_LENGTH	The input line length for record number *value* is invalid.
ERR_INVALID_PATTERN	Directory pattern '*value*' is invalid.
ERR_INVALID_PLUGIN_STATE	Invalid internal module state in *value*.
ERR_INVALID_QUEUE_SIZE	Queue size < 0.
ERR_INVALID_RECORD_LENGTH	Defined RecordLength (*value*) does not match length (*value*) of read line.
ERR_INVALID_RECORD_NUMBER	Invalid number of records: *value*, counted: *value*.
ERR_INVALID_REG_BASE_NAME	Registry base name of '*value*' does not match '*value*'.
ERR_INVALID_REG_ENTRIES	Invalid Registry Entries. *value*
ERR_INVALID_REG_VALUE	Invalid value '*value*' for '*value*'.
ERR_INVALID_REJECT_STREAM_NUMBER	Stream number is out of range.
ERR_INVALID_SEQ_NUM	Invalid sequence number '*value*'.
ERR_INVALID_SEQ_VALUE	The configuration value for *value* is invalid (*value*).
ERR_INVALID_SOCIAL_NO	Invalid social number '*value*'.
ERR_INVALID_STATE	Received EDR invalid in the current state.
ERR_INVALID_THREAD_STATE	Invalid thread state in '*value*'; *value*; *value*.
ERR_INVALID_TOKEN_COUNT	Number of HA role mediator token should be one but found '*value*'.
ERR_INVALID_TOKEN_DB_NO	Invalid HA role mediator token database number. Found '*value*' and expected to be '*value*'.
ERR_LAST_LOAD_RELOAD_FAILED	The last load/reload operation has failed.
ERR_LEN_IS_MISSING	The first item in field definition *value* must a number.
ERR_LINE_NOT_IDENTIFIED	The line couldn't identified: *value*
ERR_LINE_NOT_INSERTED_DOUBLE	Could not insert line into message DB (double key). Line *value*

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_LINE_NOT_INSERTED_INVALID	Could not insert line into message DB (invalid key). Line *value*
ERR_LINK_TABLE_INVALID	The link table *value* is invalid.
ERR_LOADING_ABORTED	Loading data aborted after *value* records.
ERR_LOADING_CUSTOMER_DATA	Loading customer data failed.
ERR_LOADING_DBTABLE	Error while loading database table *value*.
ERR_LOADING_TIMEMODEL	Loading time model failed '*value*'.
ERR_MAPPING_TABLE_INVALID	The mapping table is invalid.
ERR_MBI_INPUT_ERROR	Error while processing MBI message: *value*
ERR_MEM_MON_MEMORY_LIMIT	*value*Reached specified memory usage limit.Usage: *value* KB, available: *value* KB
ERR_MEM_MON_PROCESS_LIMIT	*value*Reached process size limit.Size: *value* KB, limit: *value* KB
ERR_MISSING_ARGUMENT	Argument '*value*' not in message '*value*'.
ERR_MISSING_LOG_FILE_NAME	Log output file name is missing.
ERR_MISSING_MESSAGE_FILE_NAME	Message file name is missing.
ERR_MISSING_REFERENCE_FIELD	Find some container references without a field reference in block *value*.
ERR_MISSING_REFERENCE_NAME	Missing reference name in block description.
ERR_MISSING_VALUES_FOR_FIELD	Find a reference field entry without id's in block *value*.
ERR_MODULE_NOT_EXIST	The module '*value*' which was configured as an event originator does not exist.
ERR_MULTIPLE_RESTART_FILES	Found more than one restart file in directory '*value*'.
ERR_NO_CLI	No **cli** in input record.
ERR_NO_CUSTOMER	No customer data for **cli** *value* in input record.
ERR_NO_CUSTOMER_DATA	No customer data present.
ERR_NO_CUSTOMER_PLUGIN	No customer plug-in present.
ERR_NO_DATABASE_PLUGIN	No database plug-in present.
ERR_NO_DEFAULT_OUTPUT_DEVICE	No default output device.
ERR_NO_DEFAULT_SENTENCE	There is no default sentence defined in the format description.
ERR_NO_DIR	Directory '*value*' not accessible.
ERR_NO_EDRFACTORY	Can't get the factory to create EDRs in *value*.
ERR_NO_EVENTHANDLER_FOUND	Event handler not found in module '*value*'.
ERR_NO_INDEX	Index '*value*' not found.
ERR_NO_MESSAGE_FILE	There are no message file found. Path : *value*
ERR_NO_ORIGINAL_RECORD	Missing the original block.
ERR_NO_PATH_NAME	No path name given.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_NO_REQUEST	Request *value* returned with no value.
ERR_NO_SEQ_VALUE	Sequence field "*value*" in sequence control file has no value.
ERR_NO_SPLITTING_PERFORMED	No splitting performed (spec-sys = *value*).
ERR_NO_SUBSCRIBER	No subscriber data for **cli** *value* in input record.
ERR_NOSP_ID_NOT_FOUND	NOSP-Id not found for Frm=*value* and AreaCode=*value*.
ERR_NOT_USABLE	The object '*value*' is not usable.
ERR_NOT_USABLE_REASON	Module is not usable: *value*.
ERR_NUMBER_OF_FIELDS_IN_RECORD	Found '*value*' instead of '*value*' fields in record '*value*'*value*.
ERR_OBJ_ALREADY_REGISTERED	'*value*' is already registered as '*value*'.
ERR_OBJ_NOT_FOUND	The object '*value*' could not be found.
ERR_OBJ_NOT_INITIALIZED	The object '*value*' is not initialized.
ERR_OBJ_NOT_REGISTERABLE	The object '*value*' could not be registered.
ERR_OBJ_NOT_REGISTERED	The object '*value*' is not registered.
ERR_OFF_MIN_GREATER_MAX	The min offset is greater than the max offset.
ERR_ONLY_ONE_EXTERNAL_DATAFIELD	There can be only one external data field for *value*.
ERR_OPEN_DIR_FAILED	Cannot open directory '*value*'; error message '*value*'.
ERR_OPEN_FILE_FAILED	Cannot open file '*value*'.
ERR_OPEN_LOG_FILE	*value*: Cannot open log file '*value*'.
ERR_OPEN_MESSAGE_FILE	*value*: Message file '*value*' could not open.
ERR_OPEN_SOCIAL_FILE	Cannot open social number file '*value*'.
ERR_OUTPUT_ALREADY_OPEN	Output stream already opened.
ERR_OUTPUT_MAPPING_FAILED	Output mapping failed: *value*.
ERR_OUTPUT_NOT_OPEN	Cannot close output stream (not open).
ERR_OUTPUT_PATH_NOT_FOUND	The output path does not exists or is not accessible.
ERR_OUTPUT_TEMP_FILE_NOT_MOVED_TO_OUTPUT	'*value*': Cannot move temporary file '*value*' to output file '*value*'.
ERR_PARAMETER_FILE_INVALID	The iRule parameter file '*value*' has an invalid format.
ERR_PARSE_DESCRIPTIONS	Failed to parse EDR description: *value*
ERR_PARSE_ERROR_DATA	Parse error on plug-in data: *value*.
ERR_PARSE_ERROR_STREAM	Parse error on input stream: *value*.
ERR_PCM_ERROR	PCM Error: err: *value* field: *value* loc *value* errclass: *value* rec_id: *value* resvd: *value* resvd2: *value- value*

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_PIPELINE_NOT_USABLE	The pipeline '*value*' is not usable; PIPELINE DEACTIVATED; check the pipeline log for error messages and start the pipeline manually.
ERR_PLUGIN_NOT_FOUND	Invalid plugin name : *value.*
ERR_PLUGIN_NOT_VALID	The module '*value*' is invalid and cannot be used.
ERR_PLUGIN_TYPE_INVALID	Module '*value*' has a wrong type.
ERR_PREFIX_DATA_NO_DELIM	Invalid delimiter count in line *value*.
ERR_PREPARE_COMMIT_TRANSACTION	Module '*value*' failed to prepare commit transaction '*value*'.
ERR_PRICE_PLUGIN_INV	Pricing data module invalid.
ERR_RATEPLAN_NOT_FOUND	Rateplan '*value*' not found in rateplan data-module.
ERR_READ_DIR_FAILED	Error reading from directory '*value*'; error message '*value*'.
ERR_READING_CONTRACT_PERIOD	Can't convert contract period length '*value*' for cli *value*.
ERR_READING_FILE	Error checking read line. Exception caught in '*value*'; *value*; *value*.
ERR_READING_SPECIALIST_SYSTEM	Can't convert specialist system number '*value*' for cli *value*.
ERR_READONLY_FILE_NOT_PROCESSED	File '*value*' is not writable, contents not processed.
ERR_REC_DESC_NOT_FOUND	Record description not found (*value*).
ERR_RECY_CANCEL_FAILED	PreRecycle: Failed to cancel transaction '*value*'. Cannot find stream name '*value*' in the recycle map.
ERR_RECY_CANNOT_SET_ITEM_TYPE	PreRecycle: Cannot set the transaction item type for transaction '*value*'.
ERR_RECY_DELETE_TMPINPUT_FILE	PreRecycle: Cannot delete temporary input file '*value*'.
ERR_RECY_FILE_NOT_INSERT	Could not insert file name '*value*' into hash table.
ERR_RECY_FILE_NOT_MOVED	The file '*value*' could not be moved to '*value*' for recycling.
ERR_RECY_FILE_OPEN	The recycle database file '*value*' can't be opening.
ERR_RECY_FILE_WRITE	Could not write to recycle database file. Try line '*value*' to insert.
ERR_RECY_ROLLBACK_FAILED	PreRecycle: Failed to rollback transaction '*value*'. Cannot move file '*value*'.
ERR_RECYTEST_FILE_NOT_COPY	The file *value* could not copy to *value* for test recycling.
ERR_REDO_POOL_ENTRY_NOT_FOUND	Redo pool entry '*value*' not found in function pool.
ERR_REFERENCENAME_NOT_IN_DEF	The reference name *value* is not in the alias description.
ERR_REG_ENTRY_NOT_FOUND	Registry entry '*value*' not found.
ERR_REG_LOCK_FILE_EXISTS	Registry lock file '*value*' already exists.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_REG_NAME_NOT_FOUND	Registry name '*value*' not found.
ERR_REG_PARSE_FAILED	Registry parse failed near '*value*'.
ERR_REG_SUBTREE_NOT_FOUND	Registry subtree '*value*' not found.
ERR_REG_UPDATE_FAILED	Command processing failed for '*value*'.
ERR_REG_VALUE_INVALID	Registry entry '*value*' has invalid value '*value*'.
ERR_REG_VALUE_IS_EMPTY	Found empty value for registry item, where a value was expected.
ERR_REJECT_STREAM_NOT_DEFINED	Reject-stream not defined in '*value*'.
ERR_RENAME_LOG_FILE	Cannot rename old logfile.
ERR_RESOLVE_STREAM_NUMBER	Failure while resolving stream number for *value*.
ERR_RETURN_PATH_NOT_FOUND	Entry for return path not found in registry.
ERR_ROLLBACK_FAILED_INPUT	Rollback failed in input-controller.
ERR_ROLLBACK_TRANSACTION	Module '*value*' failed to rollback transaction '*value*'.
ERR_SCRIPT_NOT_EXE	External program (*value*) is not executable.
ERR_SCRIPT_NOT_EXIST	Cannot find external program (*value*).
ERR_SEGMENT_NOT_DEFINED	No segment defined for `*value*`.
ERR_SEQ_ALREADY_PROCESSED	Stream with sequence number '*value*' was already processed.
ERR_SEQ_CHECK_FAILED	Sequence check failed.
ERR_SEQ_ENTRY_NOT_FOUND	Cannot find entry "*value*" in the sequence control file.
ERR_SEQ_FILE_INVALID	Error reading / parsing sequence number file '*value*'.
ERR_SEQ_GAP	Sequence number '*value*' is to high.
ERR_SEQ_INIT	Default sequence file generated. Check file content.
ERR_SEQ_MASTER_CONTROL	False master controller type in '*value*' configured.
ERR_SEQ_MASTER_CONTROLLER	Unknown or wrong master controller for sequence sharing.
ERR_SEQ_MIN_GREATER_MAX	The min sequence number is greater than the max sequence number.
ERR_SEQ_SAVE	Error saving sequence information to stream.
ERR_SETUP_CALLTYPE	Failure during setup of calltype table.
ERR_SETUP_CZT_MAPTABLE	Error while setting up CZT map table from database.
ERR_SETUP_EDRFACTORY	EDR factory setup failed: *value*.
ERR_SETUP_FSM	Failure during setup of finite state machine.
ERR_SETUP_INPUT_GRAMMAR	Input grammar setup failed: *value*
ERR_SETUP_OUTPUT	Error setup output line. Exception caught in '*value*'; *value*; *value* (line=*value*).
ERR_SETUP_OUTPUT_GRAMMAR	Output grammar setup failed: *value*

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_SHUTDOWN_FAIL_TO_COMPLETE	Shutdown request fails to finish.
ERR_SOURCE_VALUE	Source parameter must be either 'Database' or 'File'.
ERR_SPECIAL_FUNCTIONS_FAILED	The routine 'specialFunctions' in pipeline *value* failed.
ERR_STR_LEAVING_THREAD	Critical stream error. Shutting down pipeline.
ERR_STREAM_NOT_FOUND	Could not create any statistic informations for this device.
ERR_STREAM_TO_EDR_FAILED	Stream to EDR conversion failed. EDR container created, but not written to input buffer.
ERR_SYSCATALOG_ENTRY_NOT_FOUND	System catalog entry '*value*' not found
ERR_SYSTEM_ERROR	Unexpected error, *value*
ERR_TAM_ABORT_REQUESTED	Abort requested for transaction manager '*value*'.
ERR_TAM_ENTRY_NOT_FOUND	Cannot find entry "*value*" in the transaction manager map.
ERR_TAM_ENTRY_NOT_REMOVED	Cannot remove entry "*value*" from the transaction manager map.
ERR_TAM_FILE_READ_ERR	Error reading from binary transaction log file '*value*', message '*value*'.
ERR_TAM_FILE_WRITE_ERR	Error writing into binary transaction log file '*value*', message '*value*'.
ERR_TAM_INIT_FAILED	Failed to init transaction manager '*value*'.
ERR_TAM_STREAM_NOT_FOUND	Cannot find stream name "*value*" for transaction id "*value*".
ERR_TAP3_FATAL	TAP3 Fatal: Field Name: *value*, Tag: *value*, Error Code: *value*, Description: *value*
ERR_TAP3_RET	TAP3 return: Severity: *value*, Error Code: *value*, Tag: *value*, Depth: *value*, Offset: *value*, Array ID: *value*, Operator Message: *value*, Rule ID: *value*.
ERR_TAP3_SEVERE	TAP3 Severe: Field Name: *value*, Tag: *value*, Error Code: *value*, Description: *value*
ERR_TAP3_WARNING	TAP3 Warning: Field Name: *value*, Tag: *value*, Error Code: *value*, Description: *value*
ERR_TARIFF_PLUGIN_INV	Tariff model data module invalid.
ERR_TEMP_FILE_NOT_MOVED	Cannot move temporary input file '*value*'.
ERR_THREAD_EXCEPTION	Exception detected in '*value*'; *value*; *value*.
ERR_THREAD_STACKSET_FAILED	Failed to set stack size of thread
ERR_TIME_PLUGIN_INV	Time model data module invalid.
ERR_TMPFILE_NOT_MOVED	Temporary file '*value*' could not be moved to '*value*'.
ERR_TOKEN_ACCESS_TIMEOUT	Timeout while accessing HA role mediator token for read or update.
ERR_TOKEN_READ_FAILED	Failed to read HA role mediator token.
ERR_TOKEN_UPDATE_FAILED	Failed to update HA role mediator token.

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_TRACE_START_POINT_NOT_FOUND	TraceStartPoint not found.
ERR_TRACEPOINTS	TraceEndPoint is less than TraceStartPoint.
ERR_TRANS_ID_REG_ENTRY_NOT_FOUND	Registry entry '*value*' not found in transaction id information file '*value*'.
ERR_TRANS_ID_REG_INVALID_VALUE	Invalid value '*value*' for '*value*' in transaction id information file '*value*'.
ERR_TRANSFER_CUTOFF_VIOLATED	TransferCutOff Date (*value*) violated with *value*.
ERR_UNKNOWN_ALIGNMENT	Unknown alignment text in *value*. It set to left.
ERR_UNKNOWN_COL_TYPE	Unknown colType (*value*) in section info.
ERR_UNKNOWN_DEFAULT_SENTENCE	Output Stream *value* : The default line couldn't be identified. Check your format description.
ERR_UNKNOWN_DISCARD_FKT	Valid functions are [Discard or Skip]
ERR_UNKNOWN_EVENT_TYPE	Event *value* has unknown event type.
ERR_UNKNOWN_FIELD_NAME	Unknown field name (*value*) in section info.
ERR_UNKNOWN_ROW_TYPE	Unknown rowType (*value*) in section info.
ERR_UNKNOWN_SPLITTING_RULES	Unknown type of splitting rules `*value*'.
ERR_USR_PROCESS_KILLED	Killed external process '*value*' after it timed out.
ERR_VALUE_CONV_FAIL	Error converting value(s): *value*.
ERR_VERSION_CHECK_FAILED	Version check for database '*value*' and '*value*' failed.
ERR_WRITE_DEF_EDR_NOT_FOUND	Registry entry 'WriteDefaultEdr' not found.
ERR_WRITE_FILE	Cannot create/write file '*value*'.
ERR_WRONG_TOKEN_COUNT	Wrong token count in input file. Line: *value*
ERR_XML_INPUT_MAPPING_FAILED	EDR XML generation failed: Input mapping `*value*' failed: *value*.
ERR_XML_PARSE_EDR	Exception parsing XML: near Attribute: *value*: *value*
ERR_XML_PARSE_SAX	Exception parsing XML: Line: *value* Column: *value*: *value*
ERR_XML_PARSE_UNKNOWN	Unknown exception parsing XML
ERR_XML_PARSE_XML	Exception parsing XML: *value*
ERR_ZONE_PLUGIN_INV	Zone model data module invalid.
ERR_ZONEENTRY_NOT_FOUND	Cannot find entry in zone model '*value*' for origin '*value*', destin '*value*', call date '*value*' and service '*value*'.
ERR_ZONEMODEL_NOT_CONFIGURED	Zone model '*value*' has not been configured.
ERR_ZONEMODEL_NOT_FOUND	Zonemodel-Id (*value*) not found in zone data-module.
ERR_ZONETREE_NOT_FOUND	Cannot find digit tree in configuration data for zone model '*value*'.
FORMAT_DESC_IS_INCOMPLETE	The format description is incomplete (HEADER, DETAIL, TRAILER).

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
INVALID_FORMAT_DESC	The format description for '*value*' is invalid.
UNKNOW_LOGLEVEL	The specified log level is unknown. Valid values are normal, warning, minor, major and critical.
WRN_CANNOT_DETERMINE_OUTSTREAMNAME	Cannot determine the output file name for streamname '*value*'.
WRN_CCENTRY_INVALID	Invalid call class map entry: *value*.
WRN_CLI_NOT_FOUND	Cli *value* not found.
WRN_CONTRACT_NOT_FOUND	Contract *value* not found.
WRN_CZTENTRY_INVALID	Invalid CZT map entry: *value*.
WRN_DEST_CLI_NOT_FOUND	Destination cli *value* not found.
WRN_EQUAL_TARIFFIND_DATE	Both tariff indicators have same date for contract *value*.
WRN_FILE_REMOVE_OS	*value*: Cannot remove file '*value*'.
WRN_ILLEGAL_SPECIALDAYRATE	Illegal values in special dayrate *value*.
WRN_INVALID_ACTIVATED_DATE	Invalid activation date, ignoring contract *value*.
WRN_INVALID_CLI	Ignoring invalid cli *value*.
WRN_INVALID_CLI_RANGE	Ignoring invalid cli range (*value*).
WRN_INVALID_HISTORY_DATE	Contract '*value*' has an invalid history date '*value*', using '*value*'
WRN_NO_ENDTRANSACTION	A beginTransaction arrives before the endTransaction in *value*.
WRN_NO_SEQUENCE_NUMBER_ADDEDTO_TRANSACTION	No new sequencenumber generated for sequence.
WRN_NO_STREAMLOG_DEFINED	Stream log not defined.
WRN_NO_VALID_ENTRY	Entry '*value*' in file '*value*' is invalid and ignored.
WRN_REG_ENTRY_OBSOLETE	Obsolete registry entry: *value*
WRN_SEMAPHORE_NOT_PROCESSED	Semaphore was not processed; check spelling.
WRN_TXNLOGGING_OFF	Transaction logging is off, make sure that you are doing testing only!
WRN_ZONEMAP_INVALID	Invalid zone map entry: *value*.
ERR_UNLINK_FILE_ERROR	Error value while attempting to unlink of temp file: value
ERR_OPEN_FILE_ERROR	Error value while attempting to open of temp file: value
ERR_WRITE_FILE_ERROR	Error value while attempting to write of temp file: value
ERR_CLOSE_FILE_ERROR	Error value while attempting to close of temp file: value
ERR_RENAME_FILE_ERROR	Error value while attempting to rename of temp file: value
ERR_TXN_TIMEOUT	Timeout while waiting for the next request in a transaction: value

Table 73-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_RELEASE_OBJ_LOCK	Error while releasing lock for object: value
ERR_REPLENISH_POID_CACHE_FAILED	Error while processing poids: value.
ERR_PROCESS_EXIT	Attempt to exit process due to signal.
ERR_DELETION_ASS_CBD_FAILURE	Failure in deletion of ASS_CBD block.
ERR_DELETION_CP_FAILURE	Failure in deleting of CP block.
ERR_DELETION_TP_FAILURE	Failure in deleting of TP block.
ERR_CONNECT_REJECTED	Connect from 'value' rejected
ERR_INCORRECT_FILE_NAME_SPECIFICATION	Error encountered in building the output file name from the given specification: 'value' for the input file - 'value'. Defaulting to regular file naming technique for this file .
ERR_IGNORE_REGISTRY_ENTRY	Registry entry - Name : 'value' Value : 'value' is ignored value
ERR_START_OVERLOAD_DETECTION	Failed to start/restart overload detection, value.
ERR_ZONE_VALUE_NOT_FOUND	ZoneValue not found for ZM-Id=value, Date=value, SC=value, A#=value, B#=value.
ERR_SESSION_PUT_ON_HOLD	Session value is put on hold due to being passive.
ERR_SESSION_REJECTED	Session value is rejected due to being passive.

Pipeline Manager Module Error Messages

Table 73-2 lists the DAT_AccountBatch error messages.

DAT_AccountBatch

Table 73-2 DAT_AccountBatch Error Messages

Error Message	Description
ERR_ACCOUNT_DB_UPDATE_FAILED	Database update failed for account (*value*) at time (*value*).
ERR_ACCOUNTBUSY_ACCOUNT_OBJ_FIELD_NOT_FOUND	Busy account obj field not found in flist for job: *value*
ERR_ACCOUNTBUSY_BATCH_OBJ_NOT_FOUND	Batch obj not found in flist for job: *value*
ERR_ACCOUNTBUSY_BATCH_OBJ_NOT_FOUND	Batch obj not found in flist for job: *value*.
ERR_ACCOUNTBUSY_JOB_ALREADY_REMOVED	Busy job with the id does not exits: *value*.
ERR_ACCOUNTBUSY_JOB_ALREADY_REMOVED	Busy job with the id does not exits : *value*
ERR_ACCOUNTBUSY_JOB_ID_FIELD_NOT_FOUND	Busy job field not found in flist: *value*.

Table 73-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_ACCOUNTBUSY_JOB_ID_FIELD_NOT_FOUND	Busy job field not found in flist: *value*
ERR_BAD_VALUE	Null object poid in AddOrderedBalanceGroup event
ERR_BALANCE_GR_NOT_FOUND	Balance group not found for given ID
ERR_BALANCE_GROUP_UPDATE	Customer balance group update error (*value*).
ERR_BILL_INFO_UPDATE	Customer billinfo update error (*value*).
ERR_BILLING_INFO_NOT_FOUND	Did not find billing information
ERR_CUSTOMER_ACCOUNT_NOT_FOUND	Customer Account not found (*value*).
ERR_CUSTOMER_EDR_PARSING	Customer EDR parsing error (*value*).
ERR_CUSTOMER_INVALID_ITEM_POID	Customer item POID not valid (*value*).
ERR_CUSTOMER_LOGIN_ACCOUNT_NOT_FOUND	Customer account not found after login (*value*).
ERR_CUSTOMER_LOGIN_INTERNAL_ERROR	Customer login internal error (*value*).
ERR_CUSTOMER_LOGIN_NOT_FOUND	Customer login not found (*value*).
ERR_CUSTOMER_LOGIN_NOT_VALID_FOR_TIME	Customer login not valid for time (*value*).
ERR_CUSTOMER_LOGIN_SERVICE_NOT_FOUND	Customer service not found (*value*).
ERR_CUSTOMER_NO_VALID_PRODUCT	Customer charge offer not valid (*value*).
ERR_CUSTOMER_NO_VALID_PRODUCT_RATING	Customer charge offer rating not valid (*value*).
ERR_CUSTOMER_SERVICE_NOT_FOUND	Customer Service not found (*value*).
ERR_DISCOUNT_DATA_STRING	Error building discount data string for service (*value*).
ERR_DUPLICATE_ACCOUNTBUSY_JOB_ADDED	Busy job with the id already exists : (*value*).
ERR_EVENT_ORDER_MISSING_IN_MEMORY_PROFILE	EventOrder Profile object is missing for account (*value*).
ERR_EVENT_ORDER_MISSING_SCRATCH_PAD_ITEM	EventOrderImpl::doUpdateEventOrderData() cannot find the ScratchPadItem with moniker (*value*) and pipeline/transaction (*value*).
ERR_FIRST_USAGE_ITEM_ALREADY_COMMITTED	First Usage charge offer/discount offer with id (*value*). has been already committed in the pipeline.
ERR_FIRST_USAGE_OBJECT_ALREADY_INITIALIZED	First Usage charge offer/discount offer with id (*value*) has been already used and initialized in probably same or different transaction, so not initializing the validity with current time.
ERR_GET_RANGE_ITEMS	DAT_LoginDbObject::getPoidRangeItems failure: Reason=*value*
ERR_INIT_ACCOUNTS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initAccounts method - *value*

Table 73-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_INIT_BALANCE_GROUPS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initBalanceGroups method - *value*
ERR_INIT_BILL_INFOS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initBillInfo method - *value*
ERR_INIT_DELETED_ORDERED_BALANCE_GROUPS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initDeletedOrderedBalanceGroups method - *value*
ERR_INIT_GROUP_SHARING_CHARGES_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initGroupSharingCharges method - *value*
ERR_INIT_GROUP_SHARING_DISCOUNTS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initGroupSharingDiscounts method - *value*
ERR_INIT_GROUP_SHARING_PROFILES_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initGroupSharingProfiles method - *value*
ERR_INIT_LOGIN_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initLogins method - *value*
ERR_INIT_MAPPING_TABLES_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initMappingTable method - *value*
ERR_INIT_ORDERED_BALANCE_GROUPS_BILLINFO	Thread=*value* has inconsistent data in DAT::InitCustomerThread::initOrderedBalanceGroups methodfor account - *value*
ERR_INIT_ORDERED_BALANCE_GROUPS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initOrderedBalanceGroups method - *value*
ERR_INIT_PROFILES_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initProfiles method - *value*
ERR_INIT_PURCHASED_DISCOUNTSS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initPurchasedDiscounts method - *value*
ERR_INIT_PURCHASED_PRODUCTS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initPurchasedProducts method - *value*
ERR_INIT_SERVICE_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initServices method - *value*
ERR_INIT_THREAD_DIED	Thread-*value* has died with an exception in DAT_InitCustomerThread::run() - *value*
ERR_INSERTING_CUST_CREATE_EVENT_ORDER_PROFILE	Unable to insert EventOrderProfile (*value*) for newly created account (*value*) during a CustCreate event.

Table 73-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_INVALID_ENTRY_FOR_BUSINESS_PARAM	Invalid value for Business Parameter: *value* Value: *value*
ERR_INVALID_OUTPUT_STREAM	Invalid output stream (*value*).
ERR_INVALID_TYPE_CAST	Error on type cast *value*.
ERR_LISTENER_NOT_FOUND	Listener '*value*' not found.
ERR_LOOKUP_CONSTANT_ITEM	Cannot find requested item (*value*) in the ConstantItem Pool. |
ERR_MAP_MERGE_THREAD_CANCELLED	DAT_MapMergeThread::*value* has canceled because one child thread died with exception - *value*
ERR_MAP_MERGE_THREAD_DIED	DAT_MapMergeThread::*value* has died with an exception: *value*
ERR_MAPPING_TABLE_UPDATE	Customer mapping table update error (*value*).
ERR_MULTI_THREAD_INIT	DAT_InitCustomerThread failed to create and start: Reason=*value*
ERR_MULTI_THREAD_MAP_MERGE	DAT_MapMergeThread failed to create and start: Reason=*value*
ERR_NOT_ENOUGH_CONNECTIONS	Not enough connections available to coincide with the "Threads" registry value. Define a **Connections** registry value greater than or equal to the **Threads** value.
ERR_OBG_RESOLVE_ID	Error in resolving OBG Id (*value*).
ERR_REQUIRED_REG_ENTRY_MISSING	A required registry entry is missing.
ERR_REQUIRED_REG_ENTRY_NOT_CONFIGURED	Entry not configured for DAT_Account: *value*.
ERR_RERATING_IN_PROGRESS	ReRating is currently running, EDR not processed : *value*
ERR_RW_DAT_ACCOUNT_EXCEPTION	CreateScratchPadItem() failed because item with resourceHash=*value* for Pipeline/ Transaction=*value* already exists.
ERR_SCRATCH_PAD_ITEM_ALREADY_EXISTS	CreateScratchPadItem() failed because item with resourceHash=*value* for Pipeline/ Transaction=*value* already exists.
ERR_SCRATCH_PAD_ITEM_IS_READ_DIRTY	WriteData() failed because item with resourceHash=*value* has been updated by another transaction.
ERR_SCRATCH_PAD_ITEM_LOCKED_BY_ANOTHER_TRANSACTION	Unable to Read or Write ScratchPadItem because it is locked by another transaction. ResourceHash=*value* Pipeline/Transaction=*value*
ERR_SCRATCH_PAD_ITEM_NOT_FOUND	ScratchPadItem not found. ResourceHash=*value*. Pipeline Transaction Hash=*value*. FunctionName=*value*
ERR_SCRATCH_PAD_ITEM_NULL	Attempting to call AdoptAndLock() will a NULL ScratchPadItem. Pipeline Transaction Hash=*value*

Table 73-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_SCRATCH_PAD_NOT_FOUND	ScratchPad not found. Pipeline Transaction Hash=*value*. FunctionName= *value*
ERR_SERVICE_DB_UPDATE_FAILED	Account database update failed
ERR_SERVICE_NOT_CONFIGURED	Service not found (*value*).
ERR_SERVICE_OBJECT_NOT_FOUND	Service object not found for particular service Id : *value*
ERR_SERVICE_OBJECT_UPDATE_FAILED	Service object update failed for a particular service id. : *value*
ERR_SUBSCRIPTION_SERVICE_NOT_FOUND	Subscritionservicenot found (*value*).
ERR_THREAD_CANCELLED	Thread=*value* has canceled because some child thread gets an error in DAT_InitCustomerThread::run(): Info=*value* An irrecoverable error occurred in one child thread during DAT_AccountBatch multithreaded initialization. When this occurs, other nonfailing threads safely shutdown and this error message is displayed. There is no immediate resolution.
ERR_THREAD_DIED_UNEXPECTED	An irrecoverable error occurred during DAT_AccountBatch multithreaded initialization. There is no immediate resolution; instead, open a help ticket.
ERR_UNKNOWN_DAT_ACCOUNT_EXCEPTION	Unknown Exception encountered in:(*value*)
ERR_UPDATE_BILLING_STATE_BAD_DATASTRING	BillInfo::updateBillingState() failed with invalid dataStringM (*value*)
WRN_EVENT_PROCESSING_FOR_BILLINFO_FAILED	BillInfo update failed for account ID : *value*
WRN_INVALID_ACCOUNT_IN_PROFILE	Warning, one or more account profiles point to an invalid account ID.
WRN_INVALID_SERVICE_IN_PROFILE	Warning, one or more service profiles point to an invalid service ID.
WRN_MODIFY_PROFILE_NO_SERVICE	Warning, unable to complete the Modify/ CreateProfile event. Cannot locate the ERA's Service Object in the CustomerData ServiceMap. ServiceID = *value*.
WRN_MULTI_THREAD_MAP_MERGE	Login *value* found in multiple threads but update failed during map merge
WRN_REG_ENTRY_INVALID	Warning, Registry entry for DAT_Account is invalid *value*
WRN_SYSTEM_PRODUCT_MAP_IS_EMPTY	System charge offer map is empty : *value*
WRN_SYSTEM_PRODUCT_NOT_FOUND	System charge offer not found from system charge offer map for particular charge offer Id. : *value*

DAT_AccountRealtime

Table 73-3 lists the DAT_AccountRealtime error messages.

Table 73-3 DAT_AccountRealtime Error Messages

Error Message	Description
ERR_ACCRT_MESSAGE	Error message for DAT_AccountRealtime plugin module: '*value*'.
ERR_NOT_IMPLEMENTED	Method not implemented in DAT_AccountRealtime plugin module: '*value*'.

DAT_BalanceBatch

Table 73-4 lists the DAT_BalanceBatch error messages.

Table 73-4 DAT_BalanceBatch Error Messages

Error Message	Description
ERR_BALANCE_ATTACH_TRANS_MODULE	Cannot attach transaction module *value*.
ERR_BALANCE_DATABASE	Database operation failed in DAT_BalanceBatch: '*value*'
ERR_BALANCE_DETACH_MODULE	Could not detach pipeline '*value*' (not attached).
ERR_BALANCE_GET_POID_RANGE	Get poidrange failure: Reason=*value*
ERR_BALANCE_INIT_THREAD_DIED	Thread-*value* has died with an exception in DAT_InitCustomerThread::run() - *value*
ERR_BALANCE_INVALID_BALANCEDATA	Invalid balance data during '*value*'.
ERR_BALANCE_INVALID_EDRTRANSACTION	No transaction for this EDR on the transaction list in '*value*'.
ERR_BALANCE_INVALID_STATE	Invalid transaction state, transId '*value*'.
ERR_BALANCE_INVALID_TRANSACTION	Invalid transaction during '*value*'.
ERR_BALANCE_INVALID_TRANSACTIONDATA	Invalid transaction data during '*value*'.
ERR_BALANCE_LISTENER_NOT_FOUND	Listener '*value*' not found.
ERR_BALANCE_MERGE_THREAD_DIED	DAT_MapMergeThread::*value* has died with an exception: *value*
ERR_BALANCE_MESSAGE	Error message for DAT_Balance plugin module: '*value*'
ERR_BALANCE_MISSING_BALANCE_GROUP	Balance group missing.
ERR_BALANCE_PROCESS_EVENT_ERROR	Could not process event because there is an unknown error for event: *value*.
ERR_BALANCE_PROCESSING_EVENT_BEGIN	Could not begin event transaction for event id '*value*'.
ERR_BALANCE_PROCESSING_EVENT_COMMIT	Could not commit event transaction for event id '*value*'.
ERR_BALANCE_PROCESSING_EVENT_ROLLBACK	Could not rollback event transaction for event id '*value*'.
ERR_BALANCE_THREAD_DIED_UNEXPECTED	Thread=*value* has unexpectedly died: Info=*value*
ERR_BALANCE_THREAD_INIT	Initial Thread for loading balance failed to create and start: Reason=*value*

Table 73-4 (Cont.) DAT_BalanceBatch Error Messages

Error Message	Description
ERR_BALANCE_THREAD_MERGE	Merge Thread for loading balance failed to create and start: Reason=*value*
ERR_BALANCE_TRANSACTION_MISMATCH	Transactions mismatch.
ERR_BALANCE_UPDATE_BALANCE	Error while updating the balance.
WRN_BALANCE_DEADLOCK_BTN_EDRTRANS	Deadlock between edr transactions on '*value*'.
WRN_BALANCE_DEADLOCK_BTN_TRANS	Deadlock between currentpipelineId '*value*' and another pipelineId '*value*'.
WRN_BALANCE_GROUP_LOCKED	Processing on hold as BG is locked: *value*.
WRN_BALANCE_GROUP_NOT_FOUND	Balance group *value* not found.
WRN_BALANCE_INVALID_TRANSACTION	Current transaction is invalid or has errors during '*value*'.
WRN_BALANCE_INVALID_TRANSACTION_ID	Invalid transaction Id : '*value*'.
WRN_BALANCE_MERGE_THREAD	Login *value* found in multiple threads but update failed during balance merge
WRN_INVALID_CONSUMPTION_RULE	Invalid consumption rule *value* for resourceId *value* and balance group *value*

DAT_BalanceRealtime

Table 73-5 lists the DAT_BalanceRealtime error messages.

Table 73-5 DAT_BalanceRealtime Error Messages

Error Message	Description
ERR_BALRT_EXECUTING_OPCODE	Error executing the opcode: *'value'*.
ERR_BALRT_GETTING_FLIST_FIELD	Error getting flist field: *'value'*.
ERR_BALRT_MESSAGE	Error message for DAT_BalanceRealtime plugin module: *'value'*.

DAT_ConnectionManager

Table 73-6 lists DAT_ConnectionManager error messages.

Table 73-6 DAT_ConnectionManager Error Messages

Error Message	Description
ERR_ALL_SERVER_CONNECITONS_DOWN	All the server connections are down.
ERR_CREATE_LOGIN_FLIST_FAILED	Create Login flist failed .
ERR_INFRANET_GDD_INIT_FAILED	Can't initialize Infranet GDD (*value*)
ERR_INVALID_PROBE_VALUE	Incorrect probe value received for sending DPR.
ERR_LOGIN_FAILED	Login to CM (*value*) failed .
ERR_LOGIN_TO_CM	Login to CM failed for userid (*value*)

Table 73-6 (Cont.) DAT_ConnectionManager Error Messages

Error Message	Description
ERR_LOGOUT_TO_CM	Logout from CM failed for userid (*value*)
ERR_OPCODECALL_FAILED	Opcode call failed (*value*).
ERR_SERVER_CONNECT_FAILURE	Connection to server (*value*) failed, strerror is (*value*)
ERR_SERVER_CONNECTION_LOST	Server connection lost (*value*).
ERR_SERVER_RECONNECT_FAILURE	Server reconnection failed (*value*).
ERR_CLOSE_CONNECTION_FAILED	Failed to close connection for socket id: (*value*)

DAT_ConnectionPool

Table 73-7 lists the DAT_ConnectionPool error messages.

Table 73-7 DAT_ConnectionPool Error Messages

Error Message	Description
ERR_ALL_CM_CONNECITONS_DOWN	All the CM connections are down.
ERR_CM_CONNECT_FAILURE	Connection to CM (*value*) failed, strerror is (*value*)
ERR_CM_CONNECTION_LOST	CM connection lost (*value*).
ERR_CM_RECONNECT_FAILURE	CM reconnection failed (*value*).
ERR_CONNECT_FAILED	Connect call failed from CM (*value*).
ERR_CREATE_LOGIN_FLIST_FAILED	Create Login flist failed .
ERR_INFRANET_GDD_INIT_FAILED	Can't initialize Infranet GDD (*value*)
ERR_LOGIN_FAILED	Login to CM (*value*) failed .
ERR_LOGIN_TO_CM	Login to CM failed for userid (*value*)
ERR_LOGOUT_TO_CM	Logout from CM failed for userid (*value*)
ERR_OPCODECALL_FAILED	Opcode call failed (*value*).
ERR_SYSTEM_ERROR	Unexpected error (*value*).

DAT_Currency

Table 73-8 lists the DAT_Currency error message.

Table 73-8 DAT_Currency Error Messages

Error Message	Description
ERR_REGULAR_EXP	Error in Regular Expression Compilation, Desc : *value*.

DAT_Discount

Table 73-9 lists the DAT_Discount error messages.

Table 73-9 DAT_Discount Error Messages

Error Message	Description
ERR_ATTACH_DAT_DISCOUNT	Could not attach the account balance manager as '*value*' to DAT_DiscountPlugIn.
ERR_DAT_DSC_GENERIC	FATAL ERROR '*value*' line '*value*' msg '*value*' detail '*value*'.
ERR_DB_CONNECT	Database connection is invalid. Possible solution is to restart DB & send reconnect signal. Error: *value*
ERR_DETERMINE_STEP	Could not determine the step of related resource id: '*value*'.
ERR_DISCOUNT_DUPLICATE	Cannot insert new discount '*value*'.
ERR_DSC_EXCLUSION_REG_SETTING	Error in discount exclusion registry setting
ERR_DSCMISSING_DEF	'*value*'
ERR_DSCTIMEFRAME_DEF	'*value*'
ERR_EVENT_REGISTERED	Event '*value*' could not be registered to DAT_Listener.
ERR_ISCRIPT_VALIDATION_FAILED	IScript validation failed. '*value*'.
ERR_REGEXP	Invalid regular expression '*value*'.
ERR_RELOAD_EXTDATA_FAILURE	Re-Init of data in Discount Functional PlugIn or Balance Data PlugIn Failed.
ERR_RELOAD_FAILURE	Reloading discount pricing data failed.
ERR_THRESHOLDTO_SET_TO_MAX	Discount Step Threshold_To value: *value* is inappropriate, setting it to maximum: *value*"
WRN_WRONGVALUE_SET_TO_REGPARAM	Unexpected value set for registry parameter *value*.

DAT_ExchangeRate

Table 73-10 lists the DAT_ExchangeRate error message.

Table 73-10 DAT_ExchangeRate Error Message

Error Message	Description
ERR_DAT_EXCHANGERATE_INS_LIST_DB	Error in line *'value'* in database table *'value'*.

DAT_InterConnect

Table 73-11 lists the DAT_InterConnect error messages.

Table 73-11 DAT_InterConnect Error Messages

Error Message	Description
ERR_GETTING_CIBER_OCC	*value*
ERR_GETTING_NETWORK_MODEL	Unknown network model: *value*.
ERR_GETTING_NETWORK_OPERATOR	Could not get network operator for *value*.

Table 73-11 (Cont.) DAT_InterConnect Error Messages

Error Message	Description
ERR_GETTING_PRODUCT_GROUP	Could not get charge offer group for *value*.
ERR_LOADING_CIBER_OCC	Loading IFW_CIBER_OCC failed (*value*).
ERR_LOADING_ICPRODUCT	Loading IFW_ICPRODUCT failed (*value*).
ERR_LOADING_ICPRODUCT_CNF	Loading IFW_ICPRODUCT_CNF failed (*value*).
ERR_LOADING_NETWORK_MODEL	Loading IFW_NETWORKMODEL failed (*value*).
ERR_LOADING_NETWORK_OPERATOR	Loading IFW_NETWORK_OPERATOR failed (*value*).
ERR_LOADING_POI	Loading IFW_POI failed (*value*).
ERR_LOADING_SWITCH	Loading IFW_SWITCH failed (*value*).
ERR_LOADING_TRUNK	Loading IFW_TRUNK failed (*value*).
ERR_LOADING_TRUNK_CNF	Loading IFW_TRUNK_CNF failed (*value*).
ERR_SETUP_ICPRODUCT_CNF_ENTRY	Error while setting up IFW_ICPRODUCT_CNF table from database. Reason: *value*.

DAT_ItemAssign

Table 73-12 lists the DAT_ItemAssign error messages.

Table 73-12 DAT_ItemAssign Error Messages

Error Message	Description
ERR_FAILED_TO_GENERATE_MAP_TABLE	Failed to generate Tag and Type map table.
ERR_FAILED_TO_RESERVE_POID_IDS	Failed to reserve the Poid IDs.
ERR_FSM_CREATION_FAILED	Failed to get data from db or FSM creation failed *value*
ERR_INVALID_ITEM_POID_LIST	Item Poid List from DAT_Account is invalid.
ERR_NO_ITEM_TAG	Failed to get itemTag *value*
ERR_NO_TYPE_FOUND_FOR_TAG	No matching type found for given item tag.
ERR_SET_ITEM_POID_LIST_FAILED	Failed to set Item Poid List. *value*

DAT_Listener

Table 73-13 lists the DAT_Listener error messages.

Table 73-13 DAT_Listener Error Messages

Error Message	Description
ERR_CONVERTING_FLIST	FLIST string cannot be converted. *'value'*.
ERR_CONVERTING_FLIST_TO_STR	Compact FLIST cannot be converted to string. *'value'*.
ERR_OPENING_QUEUE	Error : Could not open the Queue errorCode:*value*

Table 73-13 (Cont.) DAT_Listener Error Messages

Error Message	Description
ERR_DEQUEUE_EVENT	Dequeue event exception (*'value'*).
ERR_DEQUEUE_NOT_ENABLED	Dequeuing for queue *'value'* is disabled.
ERR_ENQUEUE_EVENT	Enqueue event exception (*'value'*).
ERR_GETTING_FLIST_FIELD	Cannot get field *'value'* from FLIST.
ERR_OPENING_LOG_FILE	Fail to open the log file.
ERR_PURGE_EVENT_EXCEPT	Purging redundant events from queue *'value'* failed (exception = *'value'*).
ERR_PURGE_EVENT_RET	Purging redundant events from queue *'value'* failed (retVAlue = *'value'*).
ERR_QUEUE_NOT_FOUND	Queue *'value'* does not exist.
ERR_QUEUE_NOT_INSTALLED	Database queueing infrastructure has not been installed. This error occurs when the DAT_Listener registry value, **QueueName** does not exist in the table **user_queues**. To resolve this problem, configure the event queue.
ERR_RECEIVE_EVENT	Delivery of bus. event to DAT plugin failed (receiveEvent()).
ERR_STATVIEW_NO_ACCESS	Queue statistics view *value* cannot be accessed.
ERR_WRITING_LOG_FILE	Fail to write to the log file.
ERROR_REG_ENTRY_NOT_FOUND	Error: Registry entry not found for *value*.
WRN_NO_EVENTS	No events registered.

DAT_ModelSelector

Table 73-14 lists the DAT_ModelSelector error messages.

Table 73-14 DAT_ModelSelector Error Messages

Error Message	Description
ERR_DATABASE	FATAL ERROR '*value*' line '*value*' msg '*value*' detail '*value*'.
ERR_DETAIL_DUPLICATE	Cannot insert new detail '*value*'.
ERR_DETAIL_NULL	Cannot find detail '*value*'.
ERR_DUPLICATE_INDEX	Cannot insert new index '*value*'.
ERR_ELEM_GET_NEXT	Error getting element from the flist. Error msg : '*value*'.
ERR_GENERIC	FATAL ERROR '*value*' line '*value*' msg '*value*'.
ERR_INDEX_NOT_FOUND	Cannot find index '*value*'.
ERR_MODEL_SELECTOR_DUPLICATE	Cannot insert new model selector '*value*'.
ERR_MODELSELECTOR_LISTENER_NOT_FOUND	Listener '*value*' not found

Table 73-14 (Cont.) DAT_ModelSelector Error Messages

Error Message	Description
ERR_RELOAD_EXTDATA_FAILED	Reload of External Data Failed.
ERR_RELOAD_FAILURE	Reload Failed.
ERR_RULE_DUPLICATE	Cannot insert new rule '*value*'.
ERR_RULE_LNK_DUPLICATE	Cannot insert new rule lnk '*value*'.
ERR_RULE_NULL	Cannot find rule '*value*'.
ERR_RULE_SET_DUPLICATE	Cannot insert new rule set '*value*'.
ERR_SELECTOR_DETAIL	Selector Detail Error : *(value)*
ERR_SELECTOR_NOT_FOUND	Cannot find model selector '*value*'.
ERR_VALUE_CONV_FAIL	'*value*'.

DAT_NumberPortability

Table 73-15 lists the DAT_NumberPortability error messages.

Table 73-15 DAT_NumberPortability Error Messages

Error Message	Description
ERR_CLOSE_NP_FILE	Error closing Number Portability data file *value*.
ERR_NUM_PORT_RELOAD	Error reloading data from the Number Portability data file *value*.
ERR_NUM_PORT_DELTALOAD	Error while appending additional Number Portability data from the file *value*.

DAT_PortalConfig

Table 73-16 lists the DAT_PortalConfig error messages.

Table 73-16 DAT_PortalConfig Error Messages

Error Message	Description
ERR_CBP_DATA_TYPE_MISMATCH	Data Type mismatch for param name *value*.
ERR_CBP_GROUP_DATA_NOT_FOUND	Could not find entry for group name *value* in Map
ERR_CBP_PARAM_DATA_NOT_FOUND	Could not find entry for group name *value*, param name *value* in Map
ERR_LOADING_CBP_DATA	Could not load CBP Data.
ERR_LOADING_OOD_DATA	Could not load OOD Data.

DAT_Price

Table 73-17 lists the DAT_Price error messages.

Table 73-17 DAT_Price Error Messages

Error Message	Description
ERR_APPEND_CONFIG	Config entry could not be appended to Pricemodel-Step.
ERR_INSERT_INTO_MAP	Cannot insert entry into memory map.
ERR_INSERT_STEP	Cannot insert Pricemodel-Step onto RUM.
ERR_INVALID_GL_ACCOUNT	Current GL/Account *'value'* does not match initial value *'value'* for PM=*value*, RES=*value*, RUM=*value*.
ERR_INVALID_REVENUE_GROUP	Current RevenueGroup *'value'* does not match initial value *'value'* for PM=*value*, RES=*value*, RUM=*value*.
ERR_PRICE_MODEL_CONFIG_NOT_FOUND	Cannot find PriceModel config from the PriceModel Step object.
ERR_PRICE_MODEL_STEP_NOT_FOUND	Cannot find PriceModel step from the RUM object.
ERR_PRICEMODEL_NOT_FOUND	Cannot find PriceModel *'value'* in table IFW_PRICEMODEL.
ERR_RESOURCE_LNK_NOT_FOUND	Cannot find the ResourceLnk object.
ERR_RESOURCE_NOT_FOUND	Cannot find resource *'value'* in table IFW_RESOURCE.
ERR_RUM_NOT_FOUND	Cannot find the RUM from the ResourceLnk object.
WRN_NO_PRICEMODEL_STEPS	PM=*value* has no valid entries in IFW_PRICEMODEL_STEP configured. Skipped loading.

DAT_Rateplan

Table 73-18 lists the DAT_Rateplan error messages.

Table 73-18 DAT_Rateplan Error Messages

Error Message	Description
ERR_INVALID_RUM_IN_RUMGROUP	Found rum group(s) in IFW_RUMGROUP with no entry in IFW_RUMGROUPS_LNK ("*value*").
ERR_INVALID_RUM_IN_SERVICE	Found rum group(s) in IFW_SERVICE with no entry in IFW_RUMGROUP ("*value*").
ERR_INVALID_SPLITTING_TYPE	IFW_RATEPLAN.SPLITTING has invalid value *'value'*. Possible values are '0', '1', '2', '3'.

DAT_Recycle

Table 73-19 lists the DAT_Recycle error messages.

Table 73-19 DAT_Recycle Error Messages

Error Message	Description
ERR_QUEUE_FILE_NOT_OPENED	Error opening or creating queue file *value*.
ERR_QUEUE_FILE_READ	Error reading queue file *value*.
ERR_QUEUE_FILE_WRITE	Error writing queue file *value*.

DAT_ResubmitBatch

Table 73-20 lists the DAT_ResubmitBatch error messages.

Table 73-20 DAT_ResubmitBatch Error Messages

Error Message	Description
ERR_CREATE_TEMP_FILE	Cannot create temporary file, Error *value*.
ERR_ENQUEUE_DATA	Error enqueuing data : *value*.
ERR_INVALID_OPERATION	Operation *value*, *value*.
ERR_OPENFILE_FAILED	Cannot open file *value*, Error *value*.
ERR_PROCESS_RESUBMIT_JOB	Error occurred while processing ResubmitJob : *value*.
ERR_REMOVE_OLD_ITEMS	Error occurred while removing already Processed Items
WRN_PIPELINE_NOT_FOUND	Pipeline *value* not found for resubmitted batch *value*.
WRN_RENAME_FAILED	Cannot rename value to *value*, Error *value*.
WRN_RESUBMITINFO_NOTFOUND	ResubmitInfo not found for Pipeline *value* and File *value*.

DAT_ScenarioReader

Table 73-21 lists the DAT_ScenarioReader error messages.

Table 73-21 DAT_ScenarioReader Error Messages

Error Message	Description
ERR_DATATYPE_MISMATCH	Data type for *value* does not match value *'value'*.
ERR_GROUPING_NOT_FOUND	Grouping *value* does not exist.
ERR_INVALID_DATATYPE	Data type *value* is invalid.
ERR_INVALID_FLUSHMODE	Flushmode *value* is invalid.
ERR_INVALID_FUNCTION	Function *value* is invalid.
ERR_INVALID_VALUE	Value *'value'* is invalid.
ERR_NO_CLASSITEMS	No classitems defined for class *'value'*.

DAT_USC_Map

Table 73-22 lists the DAT_USC_Map error messages.

Table 73-22 DAT_USC_Map Error Messages

Error Message	Description
ERR_FSM_ENGINE_FAILED	FSM Engine Failed for zone model *value*.
WRN_INVALID_BITVEC_MATCH_FOR_SD	DAT_USC_Map_ZoneModel::Pattern Matching - bitVec match error for SD (*value*).
WRN_INVALID_PATH_NUM_FOR_SD	DAT_USC_Map_ZoneModel::Pattern Matching - invalid path number for SD (*value*).
WRN_INVALID_QTY_VAL	DAT_USC_Map_ZoneModel::Invalid quantity value for entry (*value*).
WRN_INVALID_TIME_FRAME	DAT_USC_Map_ZoneModel::Invalid timeframe for entry (*value*).
WRN_NO_USC_ENTRY_FOR_PATTERN.	DAT_USC_Map_ZoneModel::No usc-entries found during pattern matching.
WRN_NO_USC_ENTRY_FOR_PATTERN_AND_SD	DAT_USC_Map_ZoneModel::Pattern Matching - no usc-entries found for SD (*value*).
WRN_NO_USC_MAP_ENTRY_FOR ZONE_MODEL	DAT::USC_Map::No Usc Entries found for zone model ID (*value*).
WRN_NO_VALID_USC_ENTRY	DAT_USC_Map_ZoneModel::No Valid USC entry mapping found.

DAT_Zone

Table 73-23 lists the DAT_Zone error messages.

Table 73-23 DAT_Zone Error Messages

Error Message	Description
ERR_INSERT_ZONEMODEL	Error inserting zone model into configuration.
ERR_INV_ZONECONFIG_LINE	Error invalid zone config line: *value*.
ERR_INV_ZONECONFIG_ROW	Error invalid values in INT_STANDARD_ZONE: ZONEMODEL=*value*, ORIGIN_AREACODE=*value*, DESTIN_AREACODE=*value*, SERVICECODE=*value*, VALID_FROM=*value*, VALID_TO=*value*, ZONE_WS=*value*, ZONE_RT=*value*, ALT_ZONEMODEL=*value*.
ERR_INVALID_BEAT	Error: Invalid value for Beat (must be greater than 0).
ERR_ZONEMODEL_NOT_IN_CONFIG	Cannot find zone model *'value'* in configuration data.
WRN_INVALID_AREACODE	AreaCode contains nondigit characters. Error=*value*.

Table 73-23 (Cont.) DAT_Zone Error Messages

Error Message	Description
WRN_DUPLICATE_SERVICETYPE	Duplicate entry for ServiceType=*value* corresponding to ServiceCode=*value*.

EXT_InEasyDB

Table 73-24 lists the EXT_InEasyDB error messages.

Table 73-24 EXT_InEasyDB Error Messages

Error Message	Description
ERR_ERROR_FILE_NOT_EXIST	Jobfile '*value*' does not exist for rollback.
ERR_READING_DATA_FROM_DATABASE	Error reading data from database -> do data from database and no eof.
WRN_FILE_NOT_MOVED	Jobfile couldn't moved to actual temp-file.

EXT_PipelineDispatcher

Table 73-25 lists the EXT_PipelineDispatcher error messages.

Table 73-25 EXT_PipelineDispatcher Error Messages

Error Message	Description
ERR_WRONG_INFILEMGR	Input file manager '*value*' is not of type EXT_InFileManager
ERR_RENAME_FILE_FAILED	Failed to rename file '*value*'

FCT_Account

Table 73-26 lists the FCT_Account error messages.

Table 73-26 FCT_Account Error Messages

Error Message	Description
ERR_EMPTY_SERVICE_FIELD_MAP	No service -> edr field mapping entries for pipeline *'value'* in alias map.
ERR_JOB_RERATING_ACCOUNT	This Account is currently being Rerated.

FCT_AccountRouter

Table 73-27 lists the FCT_AccountRouter error messages.

Table 73-27 FCT_AccountRouter Error Messages

Error Message	Description
ERR_DATA_MODULE_IS_NOT_A_ROUTER	Data Module (*value*) is not configured as a Router.
ERR_DB_ROUTING_FAILED	Splitting failed (*value*).
ERR_JOB_AMT_MIGRATION	Job is under migration state and being directed.
ERR_REGISTRY_KEY_ERROR	Error found in registry key value pair (*value*).

FCT_AggreGate

Table 73-28 lists the FCT_AggreGate error messages.

Table 73-28 FCT_AggreGate Error Message

Error Message	Description
ERR_CTLFILE_NOT_CREATED	Control File *value* could not be created. Reason : *value*
ERR_DATABLOCK_NOT_FOUND	EDR datablock *'value'* not found.
ERR_EDR_ITERATOR_FAILURE	EDR indexes mismatch: *value*.
ERR_NO_DEPENDENT_CLASS_DEFINED	No dependent class defined for class *'value'* and classitem *'value'*.
ERR_SCENARIO_NOT_DEFINED	Scenario *'value'* not defined.
WRN_NO_SCENARIOS_CONFIGURED	No scenarios configured.
WRN_SCENARIO_NOT_ACTIVE	Scenario *'value'* is not active.

FCT_APN_Map

Table 73-29 lists the FCT_APN_Map error messages.

Table 73-29 FCT_APN_Map Error Messages

Error Message	Description
ERR_GPRS_GSMW_AMBIGUITY	GPRS and GSMW extensions present. This is ambiguous. (*value*).
ERR_INIT_APN_MAPTABLE	Initialize of map table failed (*value*).
ERR_INIT_EDR_ITERATOR_CHARGE_PACKET	Failed to initialize charge packet iterator (*value*).
ERR_INIT_EDR_ITERATOR_ZONE_PACKET	Failed to initialize zone packet iterator (*value*).
ERR_RAZ_MAP_NOT_USABLE	Run after zoning map table not usable (*value*).
ERR_RAZ_MAP_TABLE_NOT_INITIALISED	Run after zoning map table not initialized (*value*).
ERR_RBZ_MAP_TABLE_INVALID	Run before zoning map table not usable (*value*).
ERR_RBZ_MAP_TABLE_NOT_INITIALISED	Run before zoning map table not initialized (*value*).

FCT_ApplyBalance

Table 73-30 lists the FCT_ApplyBalance error messages.

Table 73-30 FCT_ApplyBalance Error Messages

Error Message	Description
ERR_APPLYBAL_CANCEL_DEMANDED	EDR belongs to a cancel demanded transaction.
ERR_APPLYBAL_CANCEL_EDR	EDR transaction cancel demanded.
ERR_APPLYBAL_EDR_ITERATOR	Could not reset EDR iterator.
ERR_APPLYBAL_NO_ACCOUNT_BALANCE	Could not create/find balance for BG/Resource: *value*.
ERR_APPLYBAL_NO_PREFIX	No prefix specified for the notification file name
ERR_APPLYBAL_PLUGIN_INVALID_STATE	Required action does not fit to current state '*value*'.
ERR_APPLYBAL_REALTIME	Apply Balance plugin not required for Realtime mode.
ERR_APPLYBAL_ROLLBACK_EDR	EDR transaction rollback demanded
ERR_NO_SUBBALIMPACT	No subbalance impact created for BG/Resource '*value*', discount not granted for this EDR. Make sure the resource is valid and there is a check for available resource in the configuration.

FCT_BatchSuspense

Table 73-31 lists the FCT_BatchSuspense error messages.

Table 73-31 FCT_BatchSuspense Error Messages

Error Message	Description
ERR_NO_DEFAULT_BATCH_SUSPENSE_REASON	No default batch suspense reason.
ERR_INVALID_RESUBMIT_INFO	Invalid resubmit information received for batch name *value* and pipeline *value*.
ERR_NO_BATCH_SUSPENSE_REASON	No batch suspense reason in the **/config/suspense_reason_code** object.

FCT_BillingRecord

Table 73-32 lists the FCT_BillingRecord error messages.

Table 73-32 FCT_BillingRecord Error Messages

Error Message	Description
ERR_BALANCE_NOT_FOUND	Error adding discount balance info: *value*.

FCT_CallAssembling

Table 73-33 lists the FCT_CallAssembling error messages.

Table 73-33 FCT_CallAssembling Error Messages

Error Message	Description
ERR_BAD_EDR_FILE_STREAM	std::fstream operation has failed for file: '*value*'
ERR_CALL_MISSING_FROM_DELETION_VECTOR	The rejected call '*value*' is missing from the CallDeletionVector.
ERR_CANNOT_CLEANUP_EDR_FILE	Cannot remove file: '*value*' during EDRFileManager::cleanupEDRFiles() operation.
ERR_CANNOT_CREATE_INDEX_FILE_STREAM	Unable to create the EDR index std::fstream for file: '*value*'
ERR_CANNOT_FIND_DEFAULT_OUTPUT_STREAM	Cannot lookup the defaultOutputStream during an UpgradeFlushLimit semaphore. Attempted location: *value*
ERR_CANNOT_OPEN_INDEX_FILE_STREAM	Unable to open the EDR index std::fstream for file: '*value*'
ERR_CHAIN_REFERENCE_MISSING	Chain reference is missing.
ERR_CREATING_CONTAINER_INDEX	Unable to create DETAIL container index from EDRFactory.
ERR_DELETION_MISSING_CALL_RECORD	Error, expected CallRecord missing from map during CallDeletionVector cleanup. '*value*'
ERR_DOM_EXCEPTION	DOMException caught in *value*: Message=*value*
ERR_EDR_ALREADY_CLOSED	The edr with the following chain reference is already closed: Chain reference= '*value*', LongDurationIndicator= '*value*', StartTimestamp= '*value*'
ERR_EDR_FILE_DOESNT_EXIST	EDRFile::initialize() failed because file: '*value*' does not exist.
ERR_EDR_FILE_NOT_INDEXED	The following file: '*value*' is referenced, but not available in the EDR file index.
ERR_EMPTY_MESSAGE	Reject Message *value* is missing arguments.
ERR_F_SEGMENT_ALREADY_RECEIVED	Error : An F segment has been already received with the same chain reference
ERR_FLUSH_LIMIT_IN_PROGRESS	Error: semaphore FlushLimit=*value* is already in progress, FlushLimit must finish before sending UpgradeFlushLimit.
ERR_INCLUSIVE_FLUSHZONE_LOGIC	Error InclusiveLogic, Errant Flushzone for ChainReference '*value*'.
ERR_INCLUSIVE_LOGIC_FLUSHZONE_OUT_OF_POSITION	Error InclusiveLogic, Flushzone is out-of-position.
ERR_INCLUSIVE_LOGIC_TOO_MANY_ACTIVE_CALL_SECTIONS	Error InclusiveLogic, SingleElementCallSection has too many ACTIVE CallSections.
ERR_INDEX_FILE_RENAME_FAILED	Unable to rename tmp to index file: '*value*' during IndexFile::commitFile() operation.
ERR_INVALID_CHAIN_REFERENCE	Could not find data for chain reference *value*.

Table 73-33 (Cont.) FCT_CallAssembling Error Messages

Error Message	Description
ERR_INVALID_STATE_INDICATOR	State *value* is unexpected.
ERR_INVALID_TRANSID_IN_REJECT_MSG	The transaction id sent by FCT_Reject to FCT_CallAssembling is invalid: '*value*'.
ERR_L_SEGMENT_ALREADY_RECEIVED	Error : An L segment has been already received with the same chain reference
ERR_LATEPARTIAL_EDR	Late EDR received and marked invalid; chain reference = *value*.
ERR_MISUSE_VIRTUAL_FUNCTION	Error, virtual function '*value*' is not allowed.
ERR_MULTI_CALL_SECTION_MISSING_ELEMENTS	The MultiDataElementCallSection is missing DataElements.
ERR_NO_CHAIN_REFERENCE	Chain reference is missing in message *value*.
ERR_NOT_CA_WORKFILE	File: *value* is not a Call Assembling workfile.
ERR_PRODUCING_DEFAULT_EDR	Error: cannot produce default EDR for container '*value*' during a flush operation.
ERR_RECYCLED_CALL_RECORD_MISSING	Error, CallRecord missing for recycled call, '*value*'.
ERR_RECYCLED_EDR_NOT_FOUND_IN_MAP	Error, RecycleRequest failed for chain ref '*value*'.
ERR_REJECT_CALL_RECORD_MISSING	Error, finding CallRecord during FCT_Reject AssemblyLogic lookup request. '*value*'
ERR_REJECT_CALL_SECTION_MISSING	Error, finding CallSection during FCT_Reject notification request. ChainRef='*value*' StartTime='*value*'
ERR_REJECTED_EDR_NOT_IN_WORKFILE	The rejected edr is no longer in workfile. Chain reference= '*value*', LongDurationIndicator= '*value*', StartTimestamp= '*value*'
ERR_RESTORE_ASSEMBLY_LOGIC_FAILED	Error, unable to restore AssemblyLogic in CallRecord::restore()
ERR_SAX_EXCEPTION	SAXException caught in *value*: Message=*value*
ERR_SPURIOUS_MESSAGE	Ignoring spurious message *value*.
ERR_UNABLE_TO_SERIALIZE_INDEX_ITEM	Unable to serialize an index item to the EDR index file named: '*value*'
ERR_UNKNOWN_WORKFILE_VERSION	Cannot read workfile: *value* - because of unknown version#: *value*
ERR_UNUSED_EDRS_IN_PROCESS_RESULT	Error, un-used edrs remaining in the destructed ProcessResult object.
ERR_UPGRADE_FLUSH_LIMIT_IN_PROGRESS	Error: semaphore UpgradeFlushLimit=*value* is already in progress, UpgradeFlushLimit must finish before sending FlushLimit.
ERR_UPGRADE_MODE_FAILURE	CallAssembly Error while Upgrading EDR with ChainRef=*value* and StartTime=*value*
ERR_VALID_DETAIL_DURING_FLUSH	During flush operation, the restored EDR with CHAIN_REFERENCE==*value* does not pass the isValidDetail() test.
ERR_XML_EXCEPTION	XMLException caught in *value*: Message=*value*

Table 73-33 (Cont.) FCT_CallAssembling Error Messages

Error Message	Description
ERR_XML_IMPORT_FILE_MISSING	Unable to process ImportDataFromXml semaphore because supplied XML file '*value*' is missing.
ERR_XML_MEMORY_EXCEPTION	Xerces OutOfMemoryException caught in *value*: Message=*value*

FCT_CarrierIcRating

Table 73-34 lists the FCT_CarrierIcRating error messages.

Table 73-34 FCT_CarrierIcRating Error Messages

Error Message	Description
ERR_ICPRODUCT_INVALID	No valid entry in IFW_ICPRODUCT_RATE found for NM=*value*, NO=*value*, ICPRODUCT=*value* and Date=*value*.
ERR_ICPRODUCT_NETWORKMODEL_NOT_FOUND	Network model *'value'* not found.
ERR_ICPRODUCT_NOT_FOUND	IC_PRODUCT for GROUP/TR+DIR=*value*, DATE=*value*, SNW=*value*, DNW=*value*, A#=*value*, B#=*value*, C#=*value*, RecT=*value*, SCODE=*value*, SCLASS=*value* and UC=*value* not found (Reason *'value'*).
ERR_INVALID_MODELTYPE	*value*
ERR_TRUNK_NOT_FOUND	No entry found for trunk under GSM extension.

FCT_CiberOcc

Table 73-35 lists the FCT_CiberOcc error messages.

Table 73-35 FCT_CiberOcc Error Messages

Error Message	Description
ERR_CIBEROCC_NETWORKMODEL_NOT_FOUND	The network model declared by registry parameter EdrNetworkModel cannot be found in the data module.
ERR_CIBEROCC_NOT_FOUND	A valid entry cannot be found for the source network (NM=*value*, SN=*value*, and DATE=*value*) in IFW_CIBER_OCC.
ERR_CREATE_OCC_EDR	An OCC EDR container cannot be created.

FCT_CliMapping

Table 73-36 lists the FCT_CliMapping error messages.

Table 73-36 FCT_CliMapping Error Messages

Error Message	Description
ERR_EDR_FACTORY	Error failed to get the edr factory (*value*).
WRN_CLIENTRY_INVALID	Setup cli Map entry failed (*value*).

FCT_CreditLimitCheck

Table 73-37 lists the FCT_CreditLimitCheck error messages.

Table 73-37 FCT_CreditLimitCheck Error Messages

Error Message	Description
WRN_NO_BG_OBJECT	No Balance Group Object found

FCT_CustomerRating

Table 73-38 lists the FCT_CustomerRating error messages.

Table 73-38 FCT_CustomerRating Error Messages

Error Message	Description
ERR_CUSTOMER_NOT_FOUND	No customer datablock found.
ERR_INIT_SLA_TABLE	Error during initialization from IFW_SLA table. See pipeline log for additional information.
ERR_RATEPLAN_NOT_DEFINED	No rateplan defined for customer account *'value'*.
WRN_CUSTOMER_NOT_FOUND	No customer datablock found. Using Default-Rateplan *'value'* to continue.

FCT_DataDump

Table 73-39 lists the FCT_DataDump error messages.

Table 73-39 FCT_DataDump Error Messages

Error Message	Description
ERR_INSERT_EVENT	Failed to insert event *value*

FCT_Discard

Table 73-40 lists the FCT_Discard error messages.

Table 73-40 FCT_Discard Error Messages

Error Message	Description
WRN_FIELD_NOT_FOUND	EDR field *'value'* does not exist.

FCT_Discount

Table 73-41 lists the FCT_Discount error messages.

Table 73-41 FCT_Discount Error Messages

Error Message	Description
ERR_ACCOUNT_CANCEL	Could not cancel transaction *'value'*.
ERR_ACCOUNT_COMMIT	Could not commit transaction *'value'*.
ERR_ACCOUNT_COMMIT_RESTART	Could not commit transaction *'value'* on restart.
ERR_ACCOUNT_PREPARECOMMIT	Could not prepare commit transaction *'value'*.
ERR_ACCOUNT_ROLLBACK	Could not rollback transaction *'value'*.
ERR_BEGIN_DSC_TRANSACTION	Cannot start the transaction *'value'*.
ERR_BEGIN_EDR	Cannot begin EDR transaction.
ERR_CANCEL_DEMANDED_EDR	EDR belongs to a cancel demanded transaction.
ERR_CANCEL_EDR	EDR transaction cancel demanded.
ERR_CANNOT_COMPILE_SCRIPT	Failed to compile the following IScript: *'value'*.
ERR_COMMIT_EDR	Cannot commit EDR transaction.
ERR_CURRENCY_RESID_NOT_FOUND	Failed to find this currency from DAT::Currency: *'value'*.
ERR_DISCOUNT_DETACH_MODULE	Could not detach pipeline *'value'* (not attached).
ERR_DSC_CONF_NOT_FOUND	Cannot find configuration for discount *'value'*, date *'value'*.
ERR_EDR_ITERATOR	Could not reset EDR iterator.
ERR_EDRPACK_NOT_READY_DSC	Not all EDR fields needed for discounting are filled.
ERR_END_DSC_TRANSACTION	Cannot start the transaction *'value'*.
ERR_EXPR_REF_CP	Expression referencing Charge Packet amount when there is no Charge Packet: *'value'*.
ERR_GETTING_BG_ID	Failed to get balance group id for account: *'value'*.
ERR_INVALID_BASE_AMOUNT	Invalid base amount value. Expression: *'value'*.
ERR_INVALID_BASE_EXPR	Discount with no Charge Packet (Billing Time discount) cannot reference CP amount: *'value'*.
ERR_INVALID_COND_AMOUNT	Invalid condition amount value. Expression: *'value'*.
ERR_INVALID_DISCOUNT_TYPE	Invalid discount type *'value'*.
ERR_INVALID_DRUM_AMOUNT	Invalid DRUM amount value. Expression: *'value'*.
ERR_INVALID_GRANT_TYPE	The grant type *'value'* is invalid.
ERR_INVALID_THRESHOLD_AMOUNT	Invalid threshold_to amount value. Expression: *'value'*
ERR_INVALID_THRESHOLD_TYPE	Invalid threshold type *'value'*.
ERR_NO_ACCOUNT_BALANCE	Could not create/find balance for BG/Resource: *value*.
ERR_PLUGIN_INVALID_STATE	Required action does not fit to current state *'value'*.
ERR_REJECT_EDR	EDR rejection demanded.

Table 73-41 (Cont.) FCT_Discount Error Messages

Error Message	Description
ERR_ROLLBACK_EDR	EDR transaction rollback demanded.

FCT_DroppedCall

Table 73-42 lists the FCT_DroppedCall error messages.

Table 73-42 FCT_DroppedCall Error Messages

Error Message	Description
ERR_DATA_TYPE_MISMATCH	Data type of ContinuationCallField = *value*does not match the data type of DroppedCallField = *value*
ERR_DROPPED_CALL_UPDATION_FAILED	Updation of Dropped Call entry in the memory map failed : *value*
ERR_FILE_ID_NOT_FOUND	File ID Not Found
ERR_FILE_FORMAT_INCORRECT	File Format is not correct
ERR_INSERT_INTO_MAP_FAILED	Insertion into the DroppedCallInfoMap failed : *value*
ERR_REMOVING_FILE	Could not remove file *value*
ERR_RESTORE_FAILED	Restore of the file failed
ERR_SERIALIZE_FAILED	Serialize to the file failed
ERR_UNABLE_TO_RETRIEVE_XML_ID	Unable to retrieve XML Id
ERR_VALUE_FROM_XML_ID	Unable to retrieve the value from XML Id
ERR_VALUE_PROFILE_ERA	Invalid value of DROPPED_CALL Profile ERA *value* = *value*. Profile ERA *value* is not set, So the default behavior is assumed.
WRN_FILE_NOT_FOUND	Main data file not found.

FCT_DuplicateCheck

Table 73-43 lists the FCT_DuplicateCheck error messages.

Table 73-43 FCT_DuplicateCheck Error Messages

Error Message	Description
ERR_BULKINSERTION_FAILED	Bulk Insertion Failed in the table *value* from file *value* . Error : *value.*
ERR_DELETION_FAILED	Deletion in the table *value* failed for transid :*transit_id* . Error: *value.*
ERR_DUP_IND_FIELD_TYPE	Wrong duplicate indicator field type (requires INTEGER).
ERR_FLUSH_TO_FILE	Could not flush data to file.
ERR_INSERT_MAP_FAILED	Insertion into the DupCheck Map failed from File : *value.*

Table 73-43 (Cont.) FCT_DuplicateCheck Error Messages

Error Message	Description
ERR_INSERTION_FAILED	Insertion in the table *value* failed . Error: *value.*
ERR_NO_INDEXSPACE_CONF	IndexSpaceName entry is not specified in the FCT_DuplicateCheck module in the registry. It is mandatory if the Database mode is configured.
ERR_NO_TABLESPACE_CONF	TableSpaceName entry is not specified in the FCT_DuplicateCheck module in the registry. It is mandatory if the Database mode is configured.
ERR_PROC_EXEC_FAILED	Procedure: *value* execute failed , *value.*
ERR_PROC_MISSING	Procedure: *value* does not exist.
ERR_REMOVE_FILE	Could not remove file *value.*
WRN_DUP_RECORD	Duplicate record found in the file.
WRN_EMPTY_FIELD	One key field is empty, field ignored.
WRN_MAIN_FILE_NOT_FOUND	Main data file not found.
WRN_NO_ROOT_FIELD	Defined field is not in root block.

FCT_EnhancedSplitting

Table 73-44 lists the FCT_EnhancedSplitting error messages.

Table 73-44 FCT_EnhancedSplitting Error Messages

Error Message	Description
ERR_ADD_SPLITTING_PATTERNS	Failed to add patterns to fsm (*value*): *value*.
ERR_INSERT_SPECSYS	Failed to insert '*value*=*value*' into the mapping table.
ERR_NO_SPLITTING_ENTRY	No matching splitting rule found.
ERR_NO_STREAM_FOR_SPECSYS	No output stream for specialist system *'value'*.
ERR_SETUP_SPLITTING_ENTRY	Setup for splitting entry (*value*) failed: *value*.

FCT_ExchangeRate

Table 73-45 lists the FCT_ExchangeRate error messages.

Table 73-45 FCT_ExchangeRate Error Messages

Error Message	Description
ERR_EXCHANGERATE_BRK_HEADERDATE	File creation date is invalid : *'value'*.
ERR_EXCHANGERATE_FILEDATE_NOT_EXIST	File date does not exist: *'value'*.
ERR_EXCHANGERATE_NOT_FOUND	Exchangerate not found: *'value'*,*'value'* From_Currency: *'value'*, To_Currency: *'value'* .

FCT_Filter_Set

Table 73-46 lists the FCT_Filter_Set error messages.

Table 73-46 FCT_Filter_Set Error Messages

Error Message	Description
ERR_INVALID_DISCOUNT_ID	Invalid discount object id *'value'*.
ERR_INVALID_PRODUCT_ID	Invalid charge offer id *'value'*.

FCT_IRules

Table 73-47 lists the FCT_IRules error messages.

Table 73-47 FCT_IRules Error Messages

Error Message	Description
ERR_ILLEGAL_CUSTOMER_KEY	Illegal customer key *'value'*.
ERR_ILLEGAL_LICENSE_KEY	Illegal license key *'value'*.
ERR_RULE_DESCRIPTION	Failed to create ruleset from description: *value*.
ERR_RULE_SETUP	Failed to setup rule *'value'*: *value*.
ERR_RULESET_FILE	*value*:line *value*: *value*.
ERR_RULESET_NOT_FOUND	Ruleset *'value'* not found.

FCT_IScriptPlugIn

Table 73-48 lists the FCT_IScriptPlugin error messages.

Table 73-48 FCT_IScriptPlugin Error Messages

Error Message	Description
ERR_SCRIPT_NOT_USABLE	The iScript *'value'* is not usable.

FCT_ItemAssign

Table 73-49 lists the FCT_ItemAssign error messages.

Table 73-49 FCT_ItemAssign Error Messages

Error Message	Description
ERR_INVALID_ITEM_POID	Invalid item poid returned from DAT_ItemAssign.

FCT_MainRating

Table 73-50 lists the FCT_MainRating error messages.

Table 73-50 FCT_MainRating Error Messages

Error Message	Description
ERR_INVALID_ADDON_TYPE	Addon-Type in IFW_RATEPLAN_CNF for RP=*value*, RP-V=*value*, IC=*value*, SCode=*value*, SClass=*value*, TM=*value* and TZ=*value* is invalid. Not in 'Percentage, 'Absolute or 'New value.
ERR_PRICEMODEL_CONFIG_NOT_FOUND	Pricemodel-Config not found for PM=*value*, RES=*value*, RUM=*value*, Step=*value*, Frame=*value* and Date=*value*.
ERR_PRICEMODEL_NOT_FOUND	Pricemodel *'value'* not found. See process-log-file for invalid pricemodels not loaded. Packet no. *'value'*.
ERR_PRICEMODEL_RUM_NOT_FOUND	Pricemodel-RUM not found for PM=*value*, RES=*value* and RUM=*value*. Packet no. *'value'*.
ERR_PRICEMODEL_STEP_NOT_FOUND	Pricemodel-Step not found for PM=*value*, RES=*value* RUM=*value*. Packet no. *'value'*.
ERR_RATE_PRICEMODEL_NOT_FOUND	Pricemodel not found (IFW_RATEPLAN_CNF) for RP=*value*, RPV=*value*, IC=*value*, SCode=*value*, SClass=*value*, TMM=*value* and TZ=*value*. Packet no. *'value'*.
ERR_RATEPLAN_VERSION_DATE_NOT_FOUND	Rateplan-Version not found (IFW_RATEPLAN_VER) for Rateplan=*value* and Date=*value*. Packet no. *'value'*.
ERR_RATEPLAN_VERSION_ID_NOT_FOUND	Rateplan-Version not found (IFW_RATEPLAN_VER) for Rateplan=*value* and Version=*value*. Packet no. *'value'*.
ERR_RUM_GROUP_NOT_FOUND	Found no valid rum group for packet no. *'value'*.
ERR_TIMEMODEL_NOT_FOUND	Timemodel not found (IFW_RATEPLAN_CNF) for RP=*value*, RPV=*value*, IC=*value*, SCode=*value* and SClass=*value*. Packet no. *'value'*.
ERR_TIMEZONE_NOT_FOUND	Timezone not found in TimeModel=*value* for Date=*value*. Packet no. *'value'*.

FCT_Opcode

Table 73-51 lists the FCT_Opcode error messages.

Table 73-51 FCT_Opcode Error Messages

Error Message	Description
ERR_ALL_CM_CONNECTIONS_LOST	All CM connections lost.
ERR_RECV_DATA	Error while receiving data from CM (*value*)
ERR_RESTORE_DATA	Error while restoring data (*value*)
ERR_SEND_DATA	Error occurred while sending data to CM (*value*)
ERR_SERIALIZE_EDR	Exception occurred while serializing edr (*value*)

FCT_PreRating

Table 73-52 lists the FCT_PreRating error messages.

Table 73-52 FCT_PreRating Error Messages

Error Message	Description
ERR_ASS_CBD_NOT_FOUND	No ASSOCIATED_CHARGE_BREAKDOWN block is found
ERR_NO_RATEPLAN	No rateplan-code or -id defined in DETAL.ASS_CBD.CP.
ERR_RATEPLAN_NOT_A_NUMBER	Rateplan-Id *'value'* is not a number.
ERR_RATEPLAN_TYPE_INV	Rateplan-Type *'value'* does not match valid values ('R').
ERR_RATEPLAN_VERSION_NOT_FOUND	Rateplan-Version not found for Id (*value*) and Date (*value*).
ERR_ZONE_VALUE_NOT_FOUND	ZoneValue not found for ZM-Id=*value*, Date=*value*, SC=*value*, A#=*value*, B#=*value*.

FCT_PreSuspense

Table 73-53 lists the FCT_PreSuspense error messages.

Table 73-53 FCT_PreSuspense Error Messages

Error Message	Description
ERR_DATATYPE_NOT_SUPPORTED	Data type EDR::FieldDescr::*value* is not supported.
ERR_NO_QUERYABLE_FIELDS_TABLE	No queryable fields specified in registry.
WRN_NO_QUERYABLE_FIELDS	No queryable fields specified for table *value*.

FCT_RateAdjust

Table 73-54 lists the FCT_RateAdjust error message.

Table 73-54 FCT_RateAdjust Error Message

Error Message	Description
ERR_ADD_RATEADJUST	Failure while adding rate adjust entry (*value*).

FCT_Reject

Table 73-55 lists the FCT_Reject error message.

Table 73-55 FCT_Reject Error Message

Error Message	Description
ERR_REJECT_UNKNOWN_STREAM	Error : Stream *'value'* is unknown.

FCT_Rounding

Table 73-56 lists the FCT_Rounding error message.

Table 73-56 FCT_Rounding Error Message

Error Message	Description
ERR_APP_NAME_NOT_FOUND	The Application Type name *value* provided for the Registry Entry *value* is not supported.

FCT_SegZoneNoCust

Table 73-57 list the FCT_SegZoneNoCust error message.

Table 73-57 FCT_SegZoneNoCust Error Message

Error Message	Description
ERR_INIT_SEG_ZONE_LINK	Failure during initialization of segment zone link table.

FCT_Suspense

Table 73-58 lists the FCT_Suspense error messages.

Table 73-58 FCT Suspense Error Messages

Error Message	Description
ERR_ASS_SUSPENSE_EXT_MISSING	No ASS_SUSPENSE_EXT data block. FCT_PreSuspense is not active.
ERR_EFENTRY_INVALID	Failed to setup suspense mapping entry (*value*)
ERR_NO_DEFAULT_SUSPENSE_REASON	No default suspense reason and subreason specified.
ERR_NO_MATCHING_ENTRY_IN_EDR_FLD_MAP	No matching entry found in suspense edr field map table.
ERR_REJECT_STREAM_ERROR	RejectStream registry must be set to "*value*".
WRN_FIELD_MAP_NOT_FOUND	Registry entry *'value'* not found.

FCT_Timer

Table 73-59 lists the FCT_Timer error messages.

Table 73-59 FCT_Timer Error Messages

Error Message	Description
ERR_CAN_NOT_SCHEDULE_TIMER	Can't schedule timer.
ERR_CAN_NOT_RESCHEDULE_TIMER	Can't reschedule timer id '*value*'.
ERR_CAN_NOT_RESET_TIMER	Can't reset timer id '*value*'.
ERR_CAN_NOT_CANCEL_TIMER	Can't cancel timer id '*value*'.

FCT_TriggerBill

Table 73-60 lists the FCT_TriggerBill error message.

Table 73-60 FCT_TriggerBill Error Message

Error Message	Description
ERR_TRIGGER_BILLING	TriggerBilling is required.

FCT_UoM_Map

Table 73-61 lists the FCT_UoM_Map error messages.

Table 73-61 FCT_UoM_Map Error Messages

Error Message	Description
ERR_INIT_PBC_SERVICE_MAP_TABLE	Failed to init. service map table (*value*).
ERR_INIT_PBC_SERVICE_RGL_MAP_TABLE	Failed to init. rgl map table (*value*).
ERR_INIT_UoM_MAPTABLE	Failed to intiialise the UoM map table (*value*).
ERR_NO_SERVICE_CODE_NAME_SUPPLIED	No service code value was supplied for the mapping (*value*).

FCT_UsageClassMap

Table 73-62 lists the FCT_UsageClassMap error message.

Table 73-62 FCT_UsageClassMap Error Message

Error Message	Description
ERR_UCENTRY_INVALID	Failed to setup usage class mapping entry (*value*).

FCT_USC_Map

Table 73-63 lists the FCT_USC_Map error messages.

Table 73-63 FCT_USC_Map Error Messages

Error Message	Description
ERR_SETUP_USC_MAPTABLE	Failed to initialize the USC map table (*value*).
ERR_USC_GROUP_VALUE_NOT_FOUND	No specified for USC group registry parameter.

INP_GenericStream

Table 73-64 lists the INP_GenericStream error message.

Table 73-64 INP_GenericStream Error Message

Error Message	Description
ERR_INP_GENERICSTREAM_PARSE_ERROR_STREAM	Parse error on input stream: *value*.

INP_Realtime

Table 73-65 lists the INP_Realtime error messages.

Table 73-65 INP_Realtime Error Messages

Error Message	Description
ERR_REALTIME_BLOCKINDEX_NOT_IN_FACTORY	RealtimePipeline: Container BlockIndex is missing in Factory: *value*
ERR_REALTIME_CANNOT_CREATE_DOMBUILDER	RealtimePipeline: Unable to create a DOMBuilder parser from DOMImplementationLS object.
ERR_REALTIME_CANNOT_DUPLICATE_EDR	RealtimePipeline: Unexcepted to duplicate EDR.
ERR_REALTIME_COMPILE_ISCRIPT_FAILED	RealtimePipeline: Unable to compile iscript mapping file: *value*
ERR_REALTIME_CREATE_DEFAULT_EDR_FAILED	RealtimePipeline: Unable to create a default edr for CM error propagation.
ERR_REALTIME_CREATING_CONTAINER_INDEX	RealtimePipeline: Unable to create EDR Container Index for container name: *value*
ERR_REALTIME_EDR_FIELD_MAPPING_MISMATCH	RealtimePipeline: EDRField type mismatch error, unable to map value:*value* to EDR field type: *value* in function readConstants()
ERR_REALTIME_EDR_TYPE_MISMATCH	RealtimePipeline: Type mismatch error, unable to map pin_fld_t:*value* to EDR field: *value* in function appendValueToEDR()
ERR_REALTIME_EXECUTE_ISCRIPT_FAILED	RealtimePipeline: Failed to execute realtime iscript.
ERR_REALTIME_INDEX_NOT_IN_FACTORY	RealtimePipeline: Container Index is missing in Factory: *value*
ERR_REALTIME_INSERT_FLIST_EXT	RealtimePipeline: IScript Flist extension is missing.
ERR_REALTIME_MISSING_INPUT_MAP_LIST	RealtimePipeline: XML element list <InputMap> missing in xml file: *value*

Table 73-65 (Cont.) INP_Realtime Error Messages

Error Message	Description
ERR_REALTIME_MISSING_INPUT_MAP_NODE	RealtimePipeline: XML element <InputMap> missing in xml file: *value*
ERR_REALTIME_MISSING_OPCODE_LIST	RealtimePipeline: XML element list <OpcodeMap> missing in xml file: *value*
ERR_REALTIME_MISSING_OPCODE_NODE	RealtimePipeline: XML element <OpcodeMap> missing in xml file: *value*
ERR_REALTIME_MISSING_REQUIRED_FLIST_FIELD	RealtimePipeline: The expected Input Flist field is missing: *value*
ERR_REALTIME_NULL_EDR_FACTORY	RealtimePipeline: Null EDR factory returned from edrFactory()
ERR_REALTIME_OBS_FLIST_EDR_CONVERSION_FAILED	Realtime Pipeline: Conversion of Input flist to EDR failed.
ERR_REALTIME_OPEN_ISCRIPT_FAILED	RealtimePipeline: Unable to open iscript mapping file: *value*
ERR_REALTIME_PRODUCING_EDR	RealtimePipeline: Unable to produce EDR Container from index named: *value*
ERR_REALTIME_PROPAGATE_BAD_DATABLOCK_VALUE	RealtimePipeline: PropagateBlock blockname:*value*, unable to find target EDR::DatablockValue at index [*value*,*value*]
ERR_REALTIME_PROPAGATE_BLOCK_CLONE	RealtimePipeline: PropagateBlock blockname:*value*, unable to clone source EDR::Datablock at index *value*
ERR_REALTIME_PROPAGATE_BLOCK_INVALID_ARG	RealtimePipeline: PropagateBlock blockname:*value* is not a valid candidate for propagation.
ERR_REALTIME_PROPAGATE_BLOCK_INVALID_DEPTH	RealtimePipeline: PropagateBlock blockname:*value* has an invalid depth for propagation.
ERR_REALTIME_PROPAGATE_BLOCK_LOOKUP_SOURCE_BLOCK	RealtimePipeline: PropagateBlock blockname:*value* has missing source EDR::Datablock at index *value*
ERR_REALTIME_PROPAGATE_BLOCK_MISSING	RealtimePipeline: PropagateBlock blockname:*value* is not a member of the current EDR.
ERR_REALTIME_PUSH_EDR_FAILED	RealtimePipeline: Failed to push EDRs onto InputDevice.
ERR_REALTIME_READ_REGISTRY_FAILED	RealtimePipeline: ReadRegistry() failed for INP::Realtime module.
ERR_REALTIME_REG_INTERPRETER	RealtimePipeline: IScript interpreters missing.
ERR_REALTIME_SET_INTERNAL_TRANSACION_ID_FAILED	RealtimePipeline: Failed to set Transaction Id in Internal block.
ERR_REALTIME_UNKNOWN_EXCEPTION	RealtimePipeline: Unknown exception in function: *value*

INP_Recycle

Table 73-66 lists the INP_Recycle error messages.

Table 73-66 INP_Recycle Error Messages

Error Message	Description
ERR_ADD_HEADER_REOCRD_ERROR	Error while adding header record.
ERR_ADD_TRAILER_REOCRD_ERROR	Error while adding trailer record.
ERR_INP_RECYCLE_CANNOT_CONVERT_EDR	Cannot convert EDR to new container desc, field is either missing or type has changed: *value*.
ERR_INP_RECYCLE_CANNOT_FIND_EDR_VERSION	INP::EdrVersionConverter::getEdrVersion(..) cannot find the EDR version attribute in the input xml.
ERR_INP_RECYCLE_DOM_PARSE_ERRORS	DOMParser errors encountered during parse operation in function *value*.
ERR_INP_RECYCLE_EDR_MISSING_FM	EDR xml missing FM_ELEMENT in function *value*.
ERR_INP_RECYCLE_EMPTY_QUERY_RESULT	Query for edr_fld_map_buf_t buffer returned empty data in function: *value*.
ERR_INP_RECYCLE_INVALID_READER	Invalid reader in function: *value - value.*
ERR_INP_RECYCLE_NO_DB_CONNECTION	Unable to get DB Connection in function: *value - value*.
ERR_INP_RECYCLE_NO_DB_SELECTOR	Unable to get DB Selector in function: *value - value*.
ERR_INP_RECYCLE_NO_DB_TABLES	Unable to get DB Tables edr_field_mapping_t and edr_fld_map_buf_t in function: *value.*
ERR_INP_RECYCLE_ROOT_ELEMENT_MISSING	EDR root XML element not found in function *value.*
ERR_INP_RECYCLE_SAX2_PARSE_ERRORS	SAX2XMLReader errors encountered during parse operation in function *value*.
ERR_INP_RECYCLE_UNEXPECTED_EXCEPTION	Unexpected Exception in function *value*.
ERR_PARSE_HEADER_ERROR	Error in parsing header record.
ERR_PROCESS_RECORD_ERROR	Error in processing record : *value*.
ERR_PUSH_EDR_ERROR	Error in pushing record to pipeline,record number: *value*.
ERR_RECYCLING_DATA_MODULE_NOT_FOUND	Cannot find recycling data module.
ERR_REG_TRAILER_NOT_SUPPORTED	Trailer record is not supported by INP_Recycle.

NET_EM

Table 73-67 lists the NET_EM error messages.

Table 73-67 NET_EM Error Messages

Error Message	Description
ERR_CLOSE_REALTIME_TRANSACTION	Failed to close realtime transaction in pipeline '*value*'.
ERR_CREATE_CONTEXT_FAILED	PCM_CREATE_CONTEXT failed for socket '*value*'.
ERR_LSOCK_BIND	UNIX Sock bind error for file name '*value*'.
ERR_OPCODE_NOT_CONFIGURED	Opcode '*value*' is not configured.
ERR_OPEN_REALTIME_TRANSACTION	Failed to open realtime transaction in pipeline '*value*'.
ERR_RTP_ARE_NOT_READY	All Realtime Pipelines NOT ready.
ERR_SOCK_ACCEPT	Accept failed for socket '*value*', errno '*value*'.
ERR_SOCK_BIND	TCP/IP Socket bind error '*value*', errno '*value*'.
ERR_SOCK_LISTEN	Listen failed for socket '*value*', errno '*value*'.
ERR_SOCKET_BIND	Socket bind error .

OUT_DB

Table 73-68 lists the OUT_DB error messages.

Table 73-68 OUT_DB Error Messages

Error Message	Description
ERR_DB_ROLLBACK_TRANSACTION	Database transaction collback failed for stream *'value'*.
ERR_DB_STREAM_CLOSE	Database could not closed for stream *'value'*.
ERR_DB_STREAM_OPEN	Database open failed for stream *'value'*.
ERR_FILE_NOT_CONSISTENT	Parameter *'value'* file not consistent.
ERR_INDEXLIST_NOT_CREATED	Could not create index table for edr-container fields.

OUT_GenericStream

Table 73-69 lists the OUT_GenericStream error messages.

Table 73-69 OUT_GenericStream Error Messages

Error Message	Description
ERR_OUTPUT_PARSE_ERROR	Parse error on output file: *value*.
ERR_STREAM_IS_EMPTY_RETURN	Function streamIsEmpty() needs boolean return type.

OUT_Realtime

Table 73-70 lists the OUT_Realtime error messages.

Table 73-70 OUT_Realtime Error Messages

Error Message	Description
ERR_OUT_REALTIME_CREDIT_LIMIT_CHECK_FAILED	RealtimePipeline: CreditLimitCheck failed.
WRN_OUT_REALTIME_REVERSE_RATING_APPLIED	RealtimePipeline: Reverse Rating Applied.

Pipeline Utility Error Messages

LoadIFWConfig

Table 73-71 lists the LoadIFWConfig error messages.

Table 73-71 LoadIFWConfig Error Messages

Error Message	Description
ERR_CREATE_OBJECT_FAILED	Cannot create object *value* (invalid (NULL) pointer).
ERR_INVALID_PATTERN	Directory pattern *value* is invalid.
ERR_NO_DIR	Directory *value* not accessible.
ERR_NO_PATH_NAME	No path name given.
ERR_OBJ_NOT_INITIALIZED	The object *value* is not initialized.
ERR_REG_NAME_NOT_FOUND	Registry name *value* not found.
ERR_REG_PARSE_FAILED	Registry parse failed near *value*.
ERR_REG_SUBTREE_NOT_FOUND	Registry subtree *value* not found.
ERR_REG_VALUE_IS_EMPTY	Found empty value for registry item, where a value was expected.
ERR_SYSTEM_ERROR	Unexpected error, errno *value value value*
WRN_REG_ENTRY_OBSOLETE	Obsolete registry entry: *value*

Part IX
Installing Pipeline Manager

This part describes how to install Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. It contains the following chapters:

74

Installing Pipeline Manager

This chapter describes how to install the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Installation Overview

To install and configure Pipeline Manager, follows the steps in these sections:

1. Determining Your System Requirements
2. Creating a User and Configuring Environment Variables
3. Setting the Maximum Allowed Number of Open Files
4. Installing Pipeline Manager
5. Installing Pipeline Manager Optional Components
6. Increasing Heap Size to Avoid “Out of Memory" Error Messages
7. Configuring an Oracle Pipeline Manager Database
8. (Solaris) Configuring Memory Allocation and Block Transfer Mode on Solaris Systems

Determining Your System Requirements

Before you install Pipeline Manager, ensure that requirements described in this section are met.

Hardware Requirements

Pipeline Manager requires the hardware listed in Table 74-1.

Table 74-1 Pipeline Manager Required Hardware

Component	Requirements
CPU	4 or more CPUs
RAM	More than 4 GB (to use 54-bit)
Disk Type	RAID disc array (RAID 5 hardware solution with appropriate cache strongly recommended, RAID 0 software solution)

> **Note:**
>
> The more pipelines you configure and the more CPUs you use, the more important it is to put input and output on systems that are managed by different Controllers.

Software Requirements

Pipeline Manager requires the following software:

- Oracle Database server
- Oracle Database client
- Java Runtime Environment (JRE)

See "BRM Software Compatibility" in *BRM Compatibility Matrix* for the supported software versions.

Creating a User and Configuring Environment Variables

Follow these steps to create a Pipeline Manager user and set up system environment variables on a system:

1. Create a user that owns the Pipeline Manager software. The examples in this document use the default user **integrate** with the bash shell as the default shell.

> **Note:**
>
> You must create a Pipeline Manager user and set the user's environment before installing Pipeline Manager.

2. Log in as user **integrate**.
3. Go to the *pipeline_home* directory.
4. Open the **source.me.sh** file for your shell in a text editor.

> **Note:**
>
> The **source.me** file is for a bash shell. If you use a C shell, open the **source.me.csh** file.

5. Set the following environment variables for Pipeline Manager using the information in Table 74-2:

Table 74-2 Pipeline Manager Environment Variables

Environment Variable	Description
IFW_EVENTHANDLER_PORT	The port number on which the event handler daemon listens for events. **Important:** • If you are starting more than one framework process, change the value of this parameter before starting each process. Each process must have a unique event handler port for all users on a host. A framework process will not start if it cannot start the event handler daemon, and the event handler daemon will not start if it cannot listen on the specified port. • Do not set this variable to a well-known port number, such as port 11960 used by BRM.

Table 74-2 (Cont.) Pipeline Manager Environment Variables

Environment Variable	Description
IFW_HOME	The directory where the Pipeline Manager software is installed. **Note:** This documentation refers to this directory as the *pipeline_home*.
LD_LIBRARY_PATH	The library path. Set this to include *pipeline_home***/lib**. If your system already has this variable set to another value for other applications, Pipeline Manager might not find the proper libraries. To determine if a pre-existing value is present, enter the following command: `echo $LD_LIBRARY_PATH` If the variable is already set in your system, add the new variable as follows: `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH};`*`pipeline_home`*`/lib` C shell: `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH};`*`pipeline_home`*`/lib` **Important:** After your first upgrade reset the LD_LIBRARY_PATH environment variable to the following: `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH};`*`BRM_home`*`/lib;`*`pipeline_home`*`/lib` where *BRM_home***/lib** must come before *pipeline_home***/lib**, separated by a semicolon.
LD_LIBRARY_PATH_64 (Solaris)	The library path. Set this to include *pipeline_home***/lib**. If your system already has this variable set to another value for other applications, Pipeline Manager might not find the proper 64-bit libraries. To determine if a pre-existing value is present, enter the following command: `echo $LD_LIBRARY_PATH_64` If the variable is already set in your system, add the new variable as follows: `export LD_LIBRARY_PATH_64=${LD_LIBRARY_PATH_64};`*`pipeline_home`*`/lib` C shell: `setenv LD_LIBRARY_PATH_64 ${LD_LIBRARY_PATH_64};`*`pipeline_home`*`/lib`
LD_PRELOAD_64 (Solaris)	A variable that can be used to configure libumem memory allocation. Set using the **ifw-start** script. See "libumem Memory Allocator".
MALLOC_TRIM_THRESHOLD_ **MALLOC_MMAP_MAX_** **MALLOC_TOP_PAD_** (Linux only)	On Linux Pipeline Manager platforms, modify the MALLOC_TRIM_THRESHOLD_, MALLOC_MMAP_MAX_, and MALLOC_TOP_PAD_ environment variables with these values: • MALLOC_TRIM_THRESHOLD_ = **-1** • MALLOC_MMAP_MAX_ = **0** • MALLOC_TOP_PAD_ = **268435456**
MAX28DIGIT_DECIMAL_SUPPORT	By default, Pipeline Manager supports 15-digit decimal. To configure Pipeline Manager to support up to 28-digit decimal, set this to **Y**.

Table 74-2 (Cont.) Pipeline Manager Environment Variables

Environment Variable	Description
NLS_LANG	Set this to **LANG American_America.AL32UTF8**. **Important**: You must use **American_America** as the language and territory regardless of your locale and the **UTF8 or AL32UTF8** character set. BRM 12.0 supports AL32UTF8 as its default character set. It also continues to support the UTF8 character set for BRM installations that are being upgraded to BRM 12.0 from previous versions. The unicode character set AL32UTF8 is recommended for all new BRM 12.0 deployments.
PATH	The Pipeline Manager executable path. Set this to include *pipeline_home***/bin**.
IFW_EVENTHANDLER_PORT	The port number on which the event handler daemon listens for events. **Important:** • If you are starting more than one framework process, change the value of this parameter before starting each process. Each process must have a unique event handler port for all users on a host. A framework process will not start if it cannot start the event handler daemon, and the event handler daemon will not start if it cannot listen on the specified port. • Do not set this variable to a well-known port number, such as port 11960 used by BRM.

6. Save and close the updated file.
7. Update the environment for the current shell session:

   For bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

Setting the Maximum Allowed Number of Open Files

To avoid causing a failure of Pipeline Manager, you must configure the maximum number of pipeline files allowed per process in the kernel. The recommended value is 2048. You should also set the maximum number of open files allowed for the system to 20480 (ten times the number of files per process).

To modify the number of open files allowed:

- (Linux) Setting Maximum Open Files on Linux
- (Solaris) Setting Maximum Open Files on Solaris

(Linux) Setting Maximum Open Files on Linux

To set the maximum open files on Linux:

1. Log on as **root**.
2. Open the system file (**/etc/sysctl.conf**).
3. Add the following lines to the end of the file, or modify the values if these lines are already present:

```
# sets the hard limit on file descriptors:
fs.file-max = 2605192
```

4. Run **sysctl -p** to reload all the settings from the system file.

(Solaris) Setting Maximum Open Files on Solaris

To set the maximum open files on Solaris:

1. Log on as **root**.
2. Open the system file (**/etc/system**).
3. Add the following lines to the end of the file, or modify the values if these lines are already present:

```
* sets the hard limit on file descriptors:
set rlim_fd_max = 20480
* sets the soft limit on file descriptors:
set rlim_fd_cur = 2048
```

4. Restart your computer to enable the new kernel parameters.

Installing Pipeline Manager

This section describes how to install and uninstall Pipeline Manager.

> **Note:**
>
> If you are installing Pipeline Manager to replace an identical release (for example, to restore a clean version of the package), you must first uninstall the existing installation. See "Uninstalling Pipeline Manager".

To install Pipeline Manager, see "Installing Individual BRM Components" in *BRM Installation Guide*.

To upgrade Pipeline Manager, see "Upgrading BRM and Pipeline Manager" in *BRM Upgrade Guide*.

Uninstalling Pipeline Manager

To uninstall Pipeline Manager, see "Uninstalling Optional Components" in *BRM Installation Guide*.

Pipeline Manager Directory Structure

Installation creates the following directory structure shown in Table 74-3:

> **Note:**
>
> *pipeline_home* is the default Pipeline Manager installation directory.

Table 74-3 Directory Structure Created by Pipeline Manager Installation

Directory	Description
pipeline_home/**bin/**	Server binaries, such as the main Pipeline Manager binary **ifw**.
pipeline_home/**conf/**	Contains registry files.
pipeline_home/**data/**	Contains source and directory files for EDRs.
pipeline_home/**database/**	Database creation scripts and database documentation.
pipeline_home/**discount/**	Balance Data Module transaction and data.
pipeline_home/**etc/**	Error messages. **Note:** You can copy the error messages to another directory and customize them; for example, as preparation for translation into other languages.
pipeline_home/**formatDesc/**	External file format definitions and internal container definitions.
pipeline_home/**info/**	Info registry, which shows the current system status.
pipeline_home/**instrumentation/**	Performance Data collection.
pipeline_home/**iScriptLib/**	iScript libraries.
pipeline_home/**lib/**	Shared libraries.
pipeline_home/**lib64/**	Shared libraries (64-bit)
pipeline_home/**log/**	Process log.
pipeline_home/**releaseNotes/**	Release notes.
pipeline_home/**samples/**	Sample registries that you can use for testing your Pipeline Manager setup.
pipeline_home/**semaphore/**	Directory that is checked regularly for semaphores.
pipeline_home/**tools/**	The tools such as binaries and Perl scripts.

Installing Pipeline Manager Optional Components

This section describes how to install the following Pipeline Manager optional components:

- Pipeline Configuration Manager
- Pipeline PDK Manager
- Interconnect Manager
- CIBER Roaming Manager

> **Note:**
>
> Before you install CIBER Roaming Manager and TAP Roaming Manager, you must install Interconnect Roaming Manager.

- TAP Roaming Manager

The hardware and software requirements for CIBER Roaming Manager and TAP Roaming Manager are the same as for Pipeline Manager. For more information, see "Installing Pipeline Manager".

For more information on configuring roaming in Pipeline Manager, see *BRM Configuring Roaming in Pipeline Manager*.

Before installing Pipeline Manager optional components, you must install:

- BRM.
- Pipeline Manager.

> **Note:**
>
> If you are installing the optional components to replace an identical release (for example, to restore a clean version of the package), you must first uninstall the existing installation. See "Uninstalling Pipeline Manager Optional Components".

To install Pipeline Manager optional components, see "Installing Individual BRM Components" in *BRM Installation Guide*.

Uninstalling Pipeline Manager Optional Components

To uninstall Pipeline Manager optional components, see "Uninstalling Optional Components" in *BRM Installation Guide*.

Increasing Heap Size to Avoid “Out of Memory" Error Messages

To avoid "Out of Memory" error messages in the log file after installation, increase the maximum heap size used by the Java Virtual Machine (JVM). The exact amount varies greatly with your needs and system resources. By default, the JVM used has a maximum heap size of 60 MB. Increase the maximum heap size to 120 MB by entering the following sample code in a text editor:

```
%IF_EXISTS%("INIT_JAVA_HEAP", "@INIT_JAVA_HEAP@20m") %IF_EXISTS%("MAX_JAVA_HEAP",
"@MAX_JAVA_HEAP@120m")
```

where **20m** and **120m** indicate the minimum and maximum heap sizes respectively.

Save the file as *Packagename***.ja** in the temporary directory (*temp_dir*) to which you downloaded the installation software.

Packagename indicates the name of the installation software. For example, if you installed Pipeline Manager on Solaris 11 then, save the file as **12.0_Pipeline_solaris_64_opt.ja**.

Configuring an Oracle Pipeline Manager Database

This section describes how to configure the Pipeline Manager database on Oracle.

Before proceeding, you should be familiar with executing database scripts.

Before setting up the Oracle database for Pipeline Manager, ensure the following:

- An Oracle database instance is mounted and open.

- You have administrator access to the database instance.

To install and configure the Pipeline Manager database, follow the steps in these sections:

1. Setting the Environment for the Pipeline Manager Database
2. Setting Up the Oracle JSA Database Schema
3. Setting Up the Oracle Pipeline Manager Framework Database Schema
4. Changing Public Synonyms to Private for Users in Multiuser Environments
5. Loading Procedures for FCT_DuplicateCheck
6. Encrypting the Database Password

Setting the Environment for the Pipeline Manager Database

Before setting up the database schemas for your Oracle Pipeline Manager database, you must configure database environment variables.

1. Open the configuration file for the login shell. This is the file in which your user profile is stored; for example, **.profile** or **.bashrc**. By default it is stored in your home directory.
2. Change the following variables in Table 74-4 to match your Pipeline Manager environment:

Table 74-4 Environment Variables of Pipeline Manager Database

Environment Variable	Description
ORACLE_HOME	Include or set this variable to point to the Oracle installation path.
LD_LIBRARY_PATH	(32-bit systems) Include or set entries to point to the database **/lib** directories. Example for Oracle databases: `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib32:$ORACLE_HOME/rdbms/lib32`
LD_LIBRARY_PATH_64	(64-bit systems) Include or set entries to point to the database **/lib** directories. Example for Oracle databases: `LD_LIBRARY_PATH_64=$IFW_HOME/lib:$ORACLE_HOME/lib:$ORACLE_HOME/rdbms/lib`
PATH	Include or set entries to point to the database **/bin** directory. Example for Oracle databases: `${ORACLE_HOME}/bin`

3. Save and close the file.

Setting Up the Oracle JSA Database Schema

This section describes how to run scripts to set up the Oracle JSA database schema.

Run the scripts described in the following steps to set up the database schema for the JSA tables and tablespaces:

> **Note:**
>
> All scripts are located in the *pipeline_home*/**database/Oracle/Scripts** directory.

1. Open the **JSA_Tablespaces.sql** file with a text editor.
2. Replace all occurrences of the string *** with the path to the Oracle tablespace data file.
3. Save the file.
4. Log in as user **database administration (DBA)**.
5. Run the **JSA_Tablespaces.sql** script.

   This script creates the tablespaces for the JSA server.
6. Run the **JSA_Roles.sql** script.

   This script creates roles and the default JSA user.
7. Log out and log in as user **JSA**.
8. Run the **JSA_Create.sql** script.

   This script creates the JSA tables.
9. Run the **JSA_Synonyms.sql** script.

   This script creates the JSA public synonyms.
10. Run the **JSA_Prepare.sql** script.

    This script inserts some default values into the tables.

The Oracle JSQ database schema is now set up.

Setting Up the Oracle Pipeline Manager Framework Database Schema

To set up the database schema for the Pipeline Manager framework work tables and tablespaces:

> **Note:**
>
> All scripts are located in the *pipeline_home*/**database/Oracle/Scripts** directory.

1. Open the **ifw_Tablespaces.sql** file with a text editor.
2. Replace the occurrences of the string **ORA_DAT_PATH** and **ORA_IDX_PATH** with the path that leads to the ***.dbf** files shown in the script file.
3. Save the file.
4. Log in as user **DBA**.
5. Run the **ifw_Tablespaces.sql** script.

   This script creates the tablespaces for the Pipeline Manager framework.
6. (Optional) Change the default Pipeline Manager user name (*pipeline_user*) from **integrate** to another name:

a. Open the **ifw_Roles.sql** file by using a text editor.

b. In the User: INTEGRATE section, replace the two occurrences of INTEGRATE in the following string with the new default user name:

```
INTEGRATE identified by INTEGRATE
```

c. In the GRANT commands that follows the QUOTA commands in the User: INTEGRATE section, replace all occurrences of INTEGRATE with the new default user name.

d. Save the file.

7. Run the **ifw_Roles.sql** script.

   This script creates the appropriate roles and the default user.

8. Log out and log in as *pipeline_user*.

9. Run the **ifw_Create.sql** script.

   This script creates the Pipeline Manager framework tables.

10. Run the **ifw_Synonyms.sql** script.

    This script creates the public synonyms.

The Oracle Pipeline Manager framework database schema is now set up.

You can create only one instance of the pipeline schema in a particular database. You can create multiple users and give permission to access the same tables. Creating multiple instances lead to public synonym conflict.

To avoid public synonym conflicts in a multiuser environment, change the public synonyms to private synonyms for specific users. See "Changing Public Synonyms to Private for Users in Multiuser Environments" for more information.

For example, if you create **table1** for **user1**, you can create **user2** and assign permissions to access **table1**. Typically, you access the table as **user1.table1**. However, using synonyms, you can directly access the table as table1. No prefix (**user1**) is required.

Changing Public Synonyms to Private for Users in Multiuser Environments

Changing public synonyms to private synonyms for a user is required for avoiding public synonym conflict in a multiuser environment.

> **Note:**
>
> All scripts are located in the *pipeline_home*/**database**/**Oracle**/**Scripts** directory

To change public synonyms to private synonyms for a user:

1. Connect to the Oracle database with SQL*Plus:

   ```
   % sqlplus system@databaseAlias
   Enter password: password
   ```

   where *databaseAlias* is the database alias of the Oracle database.

2. Drop the public synonyms for IFW tables for the user.

3. Grant CREATE ANY synonym access to the user you created for the Pipeline Manager tables. For example:

```
SQL> CREATE ROLE synonym_role;
SQL> GRANT CREATE ANY SYNONYM TO synonym_role;
SQL> GRANT synonym_role TO pin_user;
```

   where *pin_user* is the BRM user.

4. Open the **ifw_Synonyms.sql** file with a text editor.
5. Remove the occurrences of the string public and add the Pipeline Manager database user. For example, for ifw_SEQ_AGGREGATION table, change the string to:

```
create synonym ifw_SEQ_AGGREGATION for ifw_user.ifw_SEQ_AGGREGATION;
```

   where *ifw_user* is the is the Pipeline Manager database user.

6. Save the file.
7. Run the **ifw_Synonyms.sql** script.

   This script creates the private synonyms for the user.

8. Exit SQL*Plus.

The public synonyms are now changed to private synonyms for the user.

Loading Procedures for FCT_DuplicateCheck

Before using the FCT_DuplicateCheck module, you must load the duplicate check stored procedures in the Pipeline Manager database:

1. Load the **DuplicateCheck_Oracle.plb** file from *pipeline_home***/database/Oracle/Scripts/DuplicateCheck**.
2. Verify that there is an unique index BIDX_DUPCHK_DATA on the DATA column of the IFW_DUPLICATECHECK table. If not, then create it before starting Pipeline Manager.

The duplicate check stored procedures are now loaded.

Encrypting the Database Password

If your database is encrypted, use the **pin_crypt_upgrade** utility to migrate your data, and then use the **pin_crypt_app** utility to encrypt the Pipeline Manager password.

1. Go to the *BRM_home***/bin** directory.
2. Run the **pin_crypt_app** utility with the **-useZT** parameter:

```
pin_crypt_app -useZT -enc
```

3. At the prompt, enter the plaintext password to encrypt.
4. At the next prompt, re-enter the plaintext password.

   The output is the OZT-encrypted password.

5. Write down the encrypted password or copy it to a text editor.
6. Open the Pipeline Manager startup registry file.
7. In the **DataPool** section, enter the user name, encrypted password, and database name.

   For example:

```
UserName     = INTEGRATOR
PassWord     = &ozt|0D5E11BFDD97D2769D9B0DBFBD1BBF7EE03F1642861DFA57502C7FB85A654267
DatabaseName = IFWDB
```

> **Note:**
>
> Be sure that these lines are not commented. (The default is commented).

8. Save the registry file.

(Solaris) Configuring Memory Allocation and Block Transfer Mode on Solaris Systems

If you run Pipeline Manager on the Solaris platform, configure the following:

- libumem Memory Allocator
- Block Transfer Mode

libumem Memory Allocator

By default, the **Hoard** memory allocator is used on Solaris systems. If you run Pipeline Manager on Solaris 9, you might achieve better performance by using the **libumem** memory allocator.

To enable the **libumem** memory allocator, edit the **ifw-script** file (**ifw-start.sh** or **ifw-start.csh**, depending on your system needs) to set the **LD_PRELOAD_64** environment variable to **/usr/lib/sparcv9/libumem.so.1** and to change the executable from **ifw** to **ifw_nomalloc** as shown in the following examples:

For the **ifw-start.sh** file, change:

```
. /BRM_home/ifw/source.me.sh
ifw $*
```

To the following:

```
export LD_PRELOAD_64=/usr/lib/sparcv9/libumem.so.1
. /BRM_home/ifw/source.me.sh
ifw_nomalloc $*
```

For the **ifw-start.csh** file, change:

```
source /BRM_home/ifw/source.me.csh
ifw $*
```

To the following:

```
setenv LD_PRELOAD_64=usr/lib/sparcv9/libumem.so.1
source /BRM_home/ifw/source.me.csh
ifw_nomalloc $*
```

Block Transfer Mode

On Solaris systems, you should set up Pipeline Manager buffers to function in block transfer mode. This mode addresses performance and scalability issues that occur when two or more threads are accessing the same buffer (thread contention).

When buffers operate in block transfer mode, EDR blocks of a configurable size are transferred simultaneously between the threads. If, for example, the block size is set to 100, concurrent buffer access is reduced to 1/100 of the number without the block transfer mode.

Specifying Block Transfer Mode and Block Size

To specify block transfer mode and block size, set the values in the registry files as shown in the following example:

```
Buffer
{
  Size = 4000
  BlockTransfer = TRUE    # Optional, defaults to FALSE
  BlockSize = 500         # Only needed when BlockTransfer = TRUE
}
```

This example specifies a buffer size of 4000 and a block size of 500. Up to eight (4000/500) blocks can be in the buffer.

Loading the Tailor-Made Stored Procedure

If you use the Tailor-Made Plan feature, you must load its stored procedure.

To load the stored procedure:

1. Ensure the following:
   - BRM and Pipeline Manager are installed.
   - The BRM schema and the Pipeline Manager schema reside on the same database.
2. Connect to the Oracle database with SQL*Plus:

   ```
   % sqlplus system@databaseAlias
   Enter password: password
   ```

3. Grant access of the pipeline schema to user **pin** by doing the following:
   a. Run the SQL grant select, update, insert, and delete command on the specified Pipeline Manager tables:

      ```
      SQL> grant select, update, insert, delete on tableName to pin;
      ```

      where *tableName* is the name of the Pipeline Manager table. Run the command on the following tables:
      - ifw_rateplan
      - ifw_rateplan_cnf
      - ifw_rateplan_ver
      - ifw_model_selector
      - ifw_selector_detail
      - ifw_selector_rule

- ifw_selector_rule_lnk
- ifw_selector_ruleset
- ifw_pricemodel
- ifw_pricemdl_step

b. Run the SQL grant select command on the specified Pipeline Manager tables:

```
SQL> grant select on tableName to pin;
```

Run the command on the following tables:

- ifw_service
- ifw_timezone
- ifw_timemodel
- ifw_impact_cat
- ifw_zonemodel
- ifw_calendar

c. Run the SQL grant select command on the specified Pipeline Manager sequences:

```
SQL> grant select sequenceName to pin;
```

where *sequenceName* is the name of the Pipeline Manager sequence. Run the command on the following sequences:

- ifw_seq_selectordetail
- ifw_seq_selectorrule
- ifw_seq_modelselector
- ifw_seq_pricemodel
- ifw_seq_rateplan

4. Type **exit** to exit SQL*Plus.
5. Go to the *IFW_Home*/**database/Oracle/Scripts** directory, where *IFW_Home* is the pipeline install directory.
6. Enter the following command to open SQL*Plus:

```
% sqlplus pin@database_Name
Enter password: password
```

where *database_Name* is the service name or database alias of the Oracle database.

7. Enter the following command to load the stored procedure:

```
SQL>@create_pricing_tailormadeplan_procedures.plb
```

8. Type **exit** to exit SQL*Plus.

Loading the Discount Stored Procedure

Loading of the stored procedure is required for discount configuration.

To load the stored procedure:

1. Ensure the following:
   - BRM and Pipeline Manager are installed.

- The BRM schema and the Pipeline Manager schema are on the same database.

2. Connect to the Oracle database with SQL*Plus:

```
% sqlplus system@databaseAlias
Enter password: password
```

3. Grant access of pipeline schema to user **pin** by doing the following:

   a. Run the SQL grant select, update, insert, and delete commands on the specified Pipeline Manager tables:

   ```
   SQL> grant select, update, insert, delete on tableName to pin;
   ```

   where *tableName* is the name of the Pipeline Manager table. Run the command on the following tables:

   - ifw_discountmodel
   - ifw_dscmdl_ver
   - ifw_dscmdl_cnf
   - ifw_dsctrigger
   - ifw_dsccondition
   - ifw_discountrule
   - ifw_discountstep
   - ifw_dscbalimpact
   - ifw_discountmaster
   - ifw_discountdetail

   b. Run the SQL grant select commands on the specified Pipeline Manager sequences:

   ```
   SQL> grant select sequenceName to pin;
   ```

   where *sequenceName* is the name of the Pipeline Manager sequence. Run the command on the following sequences:

   - ifw_seq_discountmodel
   - ifw_seq_discountconfig
   - ifw_seq_discounttrigger
   - ifw_seq_discountcondition
   - ifw_seq_discountrule
   - ifw_seq_discountstep
   - ifw_seq_discountbalimpact
   - ifw_seq_discountmaster

4. Type **exit** to exit SQL*Plus.
5. Go to the **ifw/database/Oracle/Scripts** directory.
6. Enter this command to open SQL*Plus:

```
% sqlplus pin@database_Name
Enter password: password
```

   where *database_Name* is the service name or database alias of the Oracle database.

7. Enter this command to load the stored procedure:

```
SQL>@create_pricing_discountmodel_procedures.plb
```

8. Type **exit** to exit SQL*Plus.

What's Next?

You have installed Pipeline Manager and required optional components. To install Pipeline Configuration Center, see "Installing Pipeline Configuration Center" in *BRM Installation Guide*.

75
Testing Pipeline Manager

This chapter describes basic start, stop, and configuration tests for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

About Testing Pipeline Manager

To test Pipeline Manager, follow the procedures in these sections:

> **Note:**
>
> To perform a test start and stop of the Pipeline Manager database, contact your database administrator.

1. Starting Pipeline Manager
2. Testing Pipeline Manager without a Database Connection
3. Testing Pipeline Manager with a Database Connection
4. Testing Single and Multiple Pipeline Rating with BRM

Starting Pipeline Manager

You start Pipeline Manager by using one of the following methods:

- The **pin_ctl** utility.
- The **ifw** command from the *pipeline_home* directory:

  ```
  % pipeline_home/bin/ifw -r RegistryFile
  ```

  where *RegistryFile* is your registry file name.

> **Note:**
>
> When Pipeline Manager cannot establish a connection with the Pipeline Manager database (most likely because the database is down), you receive an error message and the Pipeline Manager startup is canceled.

If there are startup issues, the system stops and sends notifications to the process log and **stdout**.

> **Note:**
>
> The path and file name of the process log are defined in the **ProcessLog** section of the startup registry.

Stopping Pipeline Manager

You stop Pipeline Manager by using the **pin_ctl** utility, or a semaphore.

Testing Pipeline Manager without a Database Connection

To test Pipeline Manager *without* database access:

1. Go to the **system** directory.
2. Source the **source.me.sh** for the shell:

   ```
   % source source.me.sh
   ```

   > **Note:**
   >
   > The **source.me.sh** is for a bash shell. If you use a C shell, enter **source.me.csh**.

3. Go to the *pipeline_home* directory.
4. Start Pipeline Manager with the **simple.reg** registry file:

   ```
   bin/ifw -r pipeline_home/samples/simple/simple.reg
   ```

   The system starts without a database connection and two sample EDR files are processed.
5. To confirm that the sample EDR files are processed, go to the *pipeline_home***/samples/simple/data/out** directory and open the output file.
6. If an error occurs:
   - An output reject file is created in the *pipeline_home***/samples/simple/data/rej** directory.
   - The input file is moved to the **err** directory. You can find it in the *pipeline_home***/samples/simple/data/err** directory.
7. Stop Pipeline Manager.

Testing Pipeline Manager with a Database Connection

To test Pipeline Manager *with* database access:

1. Go to the **system** directory.
2. Source the s**ource.me.sh** for the shell:

   ```
   source source.me.sh
   ```

> **Note:**
>
> The **source.me.sh** is for a bash shell. If you use a C shell, enter **source.me.csh**.

3. Open the *pipeline_home***/samples/simple/simple.reg** file by using a text editor such as **vi**.
4. In the **ifw.DataPool.PrefixDescData** section:
   - Comment out the **Source** parameter entry with the **File** value and uncomment the entry with the **Database** value.
   - Be sure that the **DataConnection** parameter is set to **ifw.Datapool.Login**.
5. Be sure that the DBC module is configured with values for the **UserName**, **PassWord**, and **DataBaseName** parameters.
6. Save the file.
7. Start Pipeline Manager with the **simple.reg** registry file.

   ```
   bin/ifw -r pipeline_home/samples/simple/simple.reg
   ```
8. If you previously tested Pipeline Manager without a database connection, move the done EDR files from the **/samples/simple/data/done** directory to the **/samples/simple/data/out** directory and rename the file to ***edr**.

   The system is running without a database connection; and it processes two sample EDR files.
9. To confirm that the sample EDR files are processed, go to the *pipeline_home***/samples/simple/data/out** directory and open the output file.
10. If an error occurs:
    - An output reject file is created in the *pipeline_home***/samples/simple/data/rej** directory.
    - The input file is moved to the **err** directory.

      You can find it in the *pipeline_home***/samples/simple/data/err** directory.
11. Stop Pipeline Manager.

Testing Single and Multiple Pipeline Rating with BRM

This test uses the whole range of Pipeline Manager functions. To perform a **wireless** test run:

1. Go to the **system** directory.
2. Source the **source.me.sh** for the shell:

   ```
   source.me.sh
   ```

> **Note:**
>
> The **source.me.sh** is for a bash shell. If you use a C shell, enter **source.me.csh**.

3. Run the *pipeline_home***/conf/pricingdata/Oracle/insertWIRELESS_SAMPLE.pl** script.

   ```
   insertWIRELESS_SAMPLE.pl
   ```

> **Note:**
>
> If you ran this script in a previous test, you do not have to run it again.

4. Open a sample wireless registry file.

   For single pipeline testing: *pipeline_home***/conf/wireless.reg**.

> **Note:**
>
> To isolate potential problems, perform a single pipeline test first.

5. Be sure that the DBC module is configured with values for the **UserName**, **PassWord**, and **DataBaseName** parameters.
6. Start Pipeline Manager.

   For single pipeline testing, use the **wireless.reg** registry file:

   ```
   bin/ifw -r conf/wireless.reg
   ```

7. Create sample CDRs.

   See "Creating a Sample CDR File".

> **Note:**
>
> Use the file naming format **test***string***.edr**, where *string* is any string. The CDRs must match your BRM data (service, origin, timestamps).

8. Stop Pipeline Manager.

Creating a Sample CDR File

Your sample CDR must be formatted using:

- Plain ASCII
- Semi-colon-separated
- One record per line

All lines, including the last record, must end with a NL (new line) character.

Example 75-1 Sample Format:

```
service-code;a-number;b-number;start-time;duration;vol-sent;vol-recieved;callclass;cell-id;apn
```

Table 75-1 describes CDR field formats and restrictions:

Table 75-1 CDR Field Formats

Field	Description and Format
service-code	The service code. Maximum length: 3 characters. The following service code values are predefined in the sample charge: • **TEL** • **GPR** • **SMS** • **WAP** • **DAT** • **FAX**
a-number	The call's originating number. Maximum length: 40 characters. Sample value: **00491729183333**
b-number	The call's target number. Maximum length: 40 characters. Sample value: **004941067600**
start-time	The call start time. Format: YYYYMMDDHHMISS Sample values: **20011114184510** (for '14.11.2001 18:45:10')
duration	The call duration in seconds. Maximum length: 11 digits. Sample value: **300** (for 5 minutes)
vol-sent	The number of bytes sent in the call. Maximum length: 11 digits. Sample value: **(1024** for 1 KB)
vol-received	The number of bytes received in the call. Maximum length: 11 digits. Sample value: **(1024** for 1 KB)
(Optional) *callclass*	The class of call. Maximum length: 5 characters. The following call class values are predefined in the sample charge: • **Conf** = Conference Call • **Mail** = Mailbox Inquiry • **MOC** = Mobile-Originated Call (Outgoing Roaming) • **MTC** = Mobile-Terminated Call (Incoming Roaming)
apn	The access point name for the call. Maximum length: 64 characters. Sample use: specifying the URL for GPRS

Example 75-2 Sample CDR Records

```
TEL;00491729183333;004941067600;20011114184510;300;0;0;;;
TEL;00491729183333;004941067600;20011114184510;300;0;0;Mail;47113;
TEL;00491729183333;004941067600;20011114184510;300;0;0;Conf;98765;
TEL;00491729183333;004941067600;20011114184510;300;0;0;MOC;238476;
TEL;00491732410;004941067600;20011114184300;300;0;0;NORM;123456;
TEL;00491732411;004941067600;20011114184300;270;0;0;NORM;123456;
```

```
TEL;00491732412;004941067600;20011114184300;110;0;0;NORM;123456;
DAT;00491732413;004941067600;20011114184300;50;0;0;NORM;123456;
FAX;00491732414;004941067600;20011114184300;12;0;0;NORM;123456;
TEL;00491732415;004941067600;20011114184300;1;0;0;NORM;123456;
SMS;00491732416;004941067600;20011114184300;63;0;0;NORM;123456;
TEL;00491732417;004941067600;20011114184300;37;0;0;NORM;123456;
TEL;00491732418;004941067600;20011114184300;132;0;0;NORM;123456;
TEL;00491732419;004941067600;20011114184300;60;0;0;NORM;123456;
GPR;00491732410;0049;20011114184510;300;78965;5054;;001121;hamburg.portal.com
```

Troubleshooting

If you cannot start Pipeline Manager, it can be due to the following problems or errors:

> **Note:**
>
> Error messages are written into the process log file and into the pipeline log files.

- **The user environment is not set correctly.**

  Solution: Correct the errors in the environment settings.

- **The registry contains errors.**

  Solution: Check the registry for type errors, missing brackets, missing or incorrect entries, and so on.

  > **Note:**
  >
  > Registry entries are case-sensitive.

- **Paths are missing.**

  Solution: Create the missing paths according to the definition in the startup registry.

- **A lock file already exists**.

  Solution: If the BRM framework has not been stopped correctly, a lock file already exists. Delete the lock file and then stop and restart Pipeline Manager framework.

- **The database is not opened/the listener has not been started.**

  Solution: Open the database and start the listener.

- **The database entries contain errors.**

  Solution: Check the created database schemes.

Part X

Installing Account Synchronization

This part describes how to install the Oracle Communications Billing and Revenue Management (BRM) Account Synchronization feature. It contains the following chapters:

76

About Sending Account Data to Pipeline Manager

This chapter describes Oracle Communications Billing and Revenue Management (BRM) Account Synchronization Manager and how account synchronization works. Anyone who installs, configures, or administers Account Synchronization Manager should read this document.

About Account Synchronization

Account Synchronization Manager synchronizes customer and service data with pipeline rating data.

Account Synchronization Manager enables Pipeline Manager to act on events that occur in BRM based on the event type. For example, when account information changes, such as when charge offers are purchased or account status is changed, the account information must be updated in the Pipeline Manager database so that service usage events can be rated properly. The Account Synchronization DM sends the updated account information to Pipeline Manager, enabling Pipeline Manager to rate events using the updated information.

You set up the following modules to receive events through account synchronization:

- **DAT_Listener**: All events are retrieved by this module first. This module passes the events to the other modules that are set up to receive them.
- **DAT_AccountBatch**: This module receives account update information to ensure that Pipeline Manager uses the most current account information when rating events.
- **DAT_BalanceBatch**: This module receives balance update information to ensure that Pipeline Manager uses the most current balance information when rating events.
- **DAT_Discount**: This module receives discount balance update information to ensure that Pipeline Manager uses the most current discount balance configuration when rating events.

How Account Synchronization Works

The Account Synchronization DM notifies Pipeline Manager when certain account information changes (for example, when a customer service representative (CSR) adds, cancels, or modifies an account or when an adjustment such as a cycle fee is applied to an account balance). The Account Synchronization DM sends information about updates through a database queue, and Pipeline Manager retrieves the information from the queue. For more information about the queue, see "About the Database Queues".

> **Note:**
>
> When an account is created, the Account Synchronization DM notifies Pipeline Manager about the new account, but the account details (such as the account's charge offers, discount offers, and other account information) are not loaded into Pipeline Manager memory until the first call data record (CDR) is rated for the account. This is because Pipeline Manager does not require the account details until the account has some usage that must be rated.

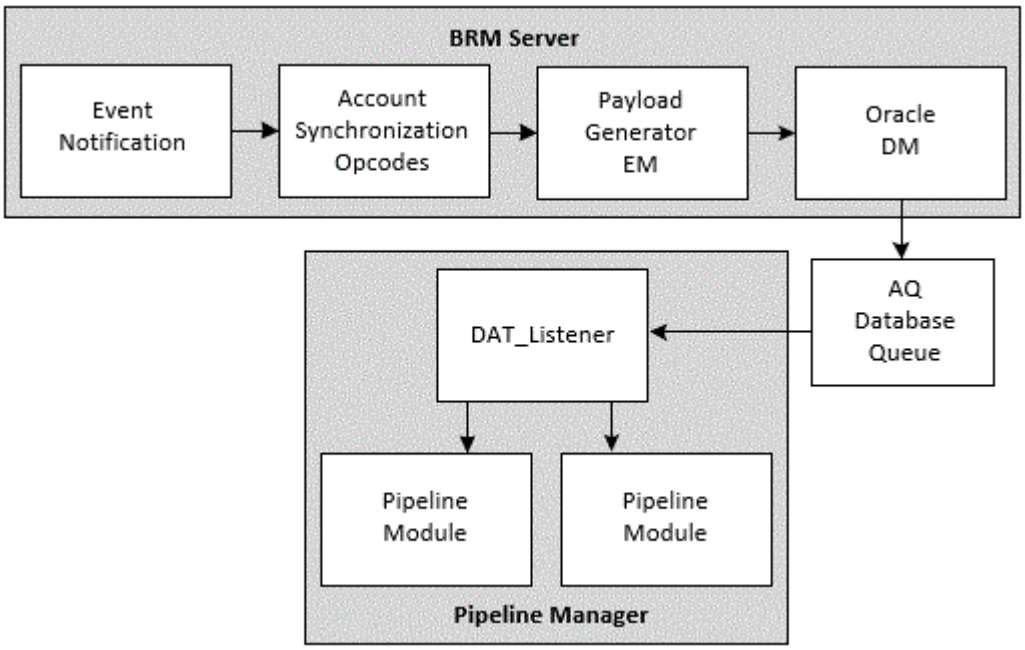
The Account Synchronization DM can send any event data to Pipeline Manager that it is configured to send. Pipeline modules register to receive events through the DAT_Listener module. When a module receives a business event, it can either use the enqueued event data for processing or access the BRM database for the most current data.

All business events sent to Pipeline Manager are stored in the database queue. If Pipeline Manager terminates, the Account Synchronization DM continues to send events to the queue. When Pipeline Manager is restarted, it retrieves the events from the queue. Likewise, if Account Synchronization Manager terminates, Pipeline Manager continues to retrieve events already in the queue.

If billing is being run while Pipeline Manager is starting, events are queued while the DAT modules are being initialized. Events are not processed until the DAT models are ready.

Figure 76-1 shows the data flow from event notification to Pipeline Manager update:

Figure 76-1 Data Flow from Event Notification to Pipeline Manager



The following actions take place when account synchronization is performed:

1. An event listed in the Account Synchronization event notification list (**pin_notify_ifw_sync**) is generated, triggering a call to the opcode associated with that event in the list.

2. One of the following actions occurs:
   - If the event is associated with the PCM_OP_IFW_SYNC_PUBLISH_EVENT opcode, it is passed to the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode for modification. The policy opcode passes the event, along with any modifications, to the Payload Generator External Module (EM).

     See "About the Account Synchronization Opcodes".
   - If the event is *not* associated with PCM_OP_IFW_SYNC_PUBLISH_EVENT, it is sent directly to the Payload Generator EM.
3. The Payload Generator EM collects events in its payload until they compose a complete business event.

   See "About the EAI Framework".
4. When the business event is complete, the Payload Generator EM sends it to the Account Synchronization DM.

   See "About the Account Synchronization DM".
5. The Account Synchronization DM sends the business event to a database queue.

   See "About the Database Queues".
6. DAT_Listener retrieves the business event from the database queue and sends the event to pipeline modules that are registered for that event.

   See "About the DAT_Listener Module".
7. The module uses the event as needed.

About the EAI Framework

The Account Synchronization DM works with the Enterprise Application Integration (EAI) framework. You use the EAI framework to define business events for account synchronization, capture the BRM events that make up the business events, and send the completed business events to the Account Synchronization DM.

The Account Synchronization EAI framework consists of the following components:

- BRM event notification

  The Account Synchronization event notification list (**pin_notify_ifw_sync**) contains all the BRM events that make up the business events defined in the Account Synchronization payload configuration file (**payloadconfig_ifw_sync.xml**). In the list, each event is associated with an opcode. When a listed event is generated, its associated opcode is called to pass the event to the Payload Generator EM.
- The Payload Generator EM

  This module notifies the Account Synchronization DM when a business event occurs that requires some action such as an account update. Default business events for account synchronization are defined in the payload configuration file.

  The **payloadconfig_ifw_sync.xml** file includes a set of default business events. If you create custom business events, use the DAT_BalanceBatch **CustomEvents** registry entry to enable Pipeline Manager to be notified of those events.

The Payload Generator EM notifies the Account Synchronization DM when business events, such as account creation, charge offer purchase, account status change, and account balance adjustment occur. As soon as the Payload Generator EM collects all the BRM events that belong to a specific business event, it sends the business event payload to the Account Synchronization DM.

Although the Account Synchronization DM relies on the EAI framework, you do not need to install EAI Manager separately. All necessary EAI files are included with Account Synchronization Manager.

About the Account Synchronization Opcodes

Account synchronization uses a set of opcodes to modify the content of certain BRM events that make up business events before the business events are sent to the Account Synchronization DM.

To enable this, the BRM events are associated with the PCM_OP_IFW_SYNC_PUBLISH_EVENT opcode in your system's event notification list. When these BRM events occur, PCM_OP_IFW_SYNC_PUBLISH_EVENT is called to pass them to the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode for processing.

You can use the policy opcode to customize certain aspects of the BRM events based on your business needs.

After the Account Synchronization opcodes process the BRM events, they pass them to the EAI framework publishing opcode, which compiles them into business events to publish to Pipeline Manager.

> **Tip:**
>
> Not all BRM events that make up Account Synchronization business events must be passed through the Account Synchronization opcodes. If you do not need to customize the BRM events, associate them with the EAI framework publishing opcode, PCM_OP_PUBLISH_GEN_PAYLOAD (number 1301), in your system's event notification list instead. This opcode is internal to BRM.

About the Account Synchronization DM

The Account Synchronization DM sends business events to the database queue. You define how the DM connects to the database queue and which business events to send to the queue through two configuration files:

- The Account Synchronization DM configuration file (**pin.conf**) specifies how to connect to the queuing database schema.

  In a multischema system, to send events to multiple queues in different schemas, you must grant each source schema user execute permission for **acct_sync** from the target schema. The Account Synchronization DM connects to only one database schema and uses schema qualifications to send events to queues in other database schemas.

- The **ifw_sync_queuenames** file specifies the business events to send to your database queues.

> **Note:**
>
> If your system contains multiple schemas, you must install and configure an Account Synchronization DM for the primary schema. Optionally, you can also install an Account Synchronization DM for each secondary schema.

When the Account Synchronization DM receives a business event, it does the following:

1. Determines which queue to send the event to by checking the **ifw_sync_queuenames** file.
2. Enqueues the event in the appropriate queue.
3. Sets the event in the queue to a READY state.

   For information, see "About Event Status Flags".

About Disconnecting the Account Synchronization DM from the Queue

You can prevent the Account Synchronization DM from enqueuing events to the database queue by using the **pin_ctl** utility. This enables you to make changes to the queuing database schema without affecting the account synchronization process.

You control the connection between the Account Synchronization DM and the database queue by using the **pin_ctl** utility.

For information about keeping Pipeline Manager online when you tune or shut down the queuing database, see "About Disconnecting DAT_Listener from the Queue".

About the Database Queues

Account synchronization uses a persistent database queue to pass business events from the Account Synchronization DM to Pipeline Manager. The queue enables account synchronization to pass events asynchronously, so BRM and Pipeline Manager are not required to be running at the same time.

Creating Database Queues

The Account Synchronization DM installer automatically creates a default queue in a specified database schema. If your system requires multiple queues, you must create additional queues by manually running the **pin_ifw_sync_oracle** utility.

> **Note:**
>
> To avoid system errors, do not run Pipeline Manager while you run the **pin_ifw_sync_oracle** utility.

When you create a queue, you must decide the following:

- The database schema in which to create the queue.

  You can create the queue in your BRM database, in the Pipeline Manager database, or in its own separate database.
- The tablespace in which to create the queue.

  You can create the queue in an existing tablespace or in its own separate tablespace.

> **Note:**
>
> For optimal performance in production systems, create the queue in its own tablespace in the Pipeline Manager database.

- The number of queues to create.

  The number of queues to create depends on your system's configuration. Each Pipeline Manager instance connects to its own database queue. Therefore, if your system contains only one instance of Pipeline Manager, you create only one database queue; if your system contains two instances of Pipeline Manager, you create two database queues.

  In multischema systems, each BRM database schema has one corresponding instance of Pipeline Manager and one corresponding database queue.

About Event Status Flags

Events in the Oracle Advanced Queuing (AQ) queue are set to the following states:

- READY indicates that the event has not been dequeued and processed by DAT_Listener.
- PROCESSED indicates that the event was dequeued by DAT_Listener. The Oracle Queue Monitor process (QMn) removes the event from the queue after a configurable amount of time.

You can check the status of events in your queue by running a report.

About the DAT_Listener Module

Pipeline Manager uses DAT_Listener to retrieve business events from the database queue. When Pipeline Manager is running, DAT_Listener continuously checks the queue for events to process.

When DAT_Listener finds an event to process, it does the following:

1. Dequeues the event from the database queue.
2. Sets the event in the queue to a PROCESSED state.
3. Determines where to route the event by using the Pipeline Manager registry.
4. Sends the event to the appropriate pipeline module.

On Oracle AQ systems, DAT_Listener can control whether Pipeline Manager processes business events or CDRs by interleaving the two processes. You can configure DAT_Listener for concurrent or interleaved processing. See "About Controlling the Business Event Backlog".

About Using Multiple Threads to Dequeue Events

By default, DAT_Listener uses one thread for dequeuing events. All events in the queue are processed in the order in which they are queued.

You can generate additional threads by using the DAT_Listener EventThreadAllocation registry entry:

- You can generate separate threads for processing different business events. This can enhance performance when the type of business event, such as RecycleRequest business events, takes longer to process.

- You can generate additional threads for the same type of business event. This can further enhance performance when there are multiple business events of the same type that are queued in succession.

A new thread can begin processing an event as soon as processing for the previous event has begun.

For example, the following registry entry will generate four threads: one thread for RecycleRequest business events, two threads for OpenNewActgCycle business events, and one default thread (for which no entry is required) for all other types of events:

```
EventThreadAllocation
{
  RecycleRequest = 1
  OpenNewActgCycle = 2
}
```

Based on this registry configuration, if the following events are queued in this order:

1. RecycleRequest #1
2. CycleForward #1
3. OpenNewActgCycle #1
4. OpenNewActgCycle #2
5. OpenNewActgCycle #3
6. OpenNewActgCycle #4
7. CycleForward #2
8. RecycleRequest #2
9. CycleForward #3

They are processed during account synchronization in this order:

1. The RecycleRequest thread starts processing RecycleRequest event #1.
2. The default thread starts processing CycleForward event #1.
3. Two OpenNewActgCycle threads process OpenNewActgCycle events #1 through #4, in order.

   For example, thread 1 processes event #1 and thread 2 processes event #2. Whichever thread finishes first processes event #3, and so on.
4. When processing starts for OpenNewActgCycle event #4, the default thread starts processing CycleForward event #2 (assuming it has completed CycleForward event #1).
5. When processing starts for CycleForward event #2, the RecycleRequest thread starts processing RecycleRequest event #2 (assuming it has completed RecycleRequest event #1).
6. When processing starts for RecycleRequest event #2, the default thread starts processing CycleForward event #3 (assuming it has completed CycleForward event #2).

You can allocate only one thread for recycle requests.

Dequeuing in Batches

Dequeuing business events from the Account Synchronization queue takes longer than sending them to Pipeline Manager. To shorten the time it takes to process business events,

DAT_Listener dequeues business events in batches. It then sends one event at a time to Pipeline Manager.

DAT_Listener retrieves events in batches only for events that require account synchronization, such as account data changes. These are the most frequent types of business events sent through the queue. Other events—for example, those that invoke a process such as a recycling a job—continue to be dequeued one at a time.

If a business event in a batch of dequeued events contains an error, all events in the batch are rolled back in BRM and Pipeline Manager is shut down. Shutting down Pipeline Manager clears the batch of business events from the pipeline memory. After you correct the error and stop and restart Pipeline Manager, BRM resends the business events to Pipeline Manager.

About Disconnecting DAT_Listener from the Queue

You can keep Pipeline Manager online when you tune or shut down the queuing database by disconnecting DAT_Listener from the queue. You can then reconnect the module to the queue after you finish making changes to the database.

When you *disconnect* DAT_Listener, it performs the following actions:

1. Waits for the total counter of active threads to become 0.
2. Disconnects each active thread:
   - If a thread is currently processing an event, DAT_Listener waits for the event to finish, inactivates the thread, and decrements the counter.
   - If a thread is not currently processing an event, DAT_Listener inactivates the thread and decreases the counter.
3. When the counter becomes 0, DAT_Listener wakes up from the wait state.

   It returns to the caller that all connections between it and the database queue have been terminated.

When you *reconnect* DAT_Listener, it reconnects to the queuing database and activates all threads.

You control the connection between DAT_Listener and the database queue by using DAT_Listener semaphores.

About Account Synchronization in a Multischema System

When you use a multischema system, you set up an instance of Pipeline Manager for each database schema. On one instance of Pipeline Manager, you configure the multischema account router module. This module keeps track of which database schema each account belongs to.

To set up account synchronization for multiple database schemas, you install and configure an Account Synchronization DM on the primary BRM installation machine and, optionally, on your secondary BRM installation machines. You also do the following:

- **Add a database queue for each BRM database schema.**

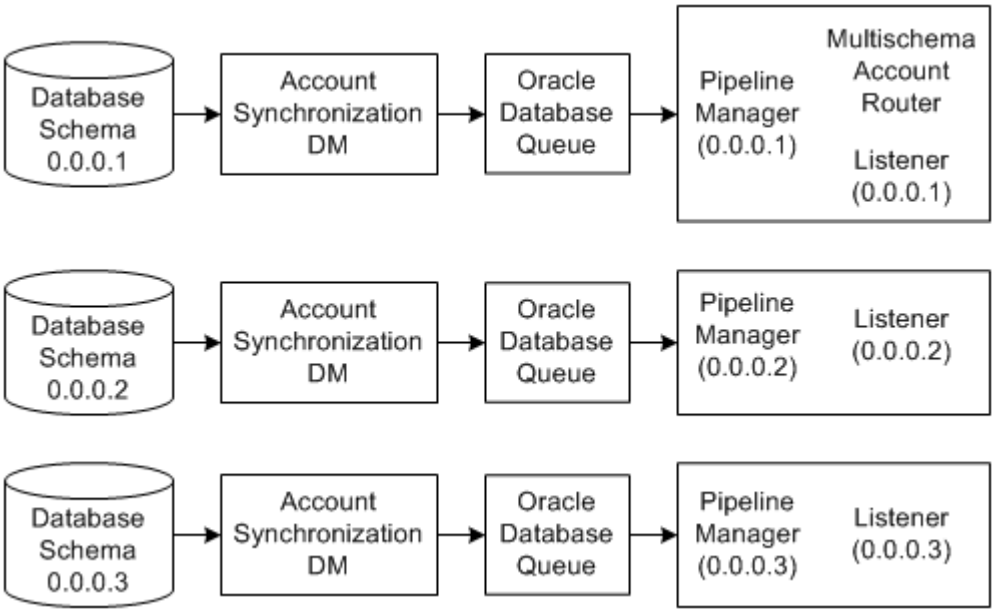
  Each Account Synchronization DM connects to only one database schema and uses schema qualifications to put event messages in queues in other database schemas.
- **Configure an instance of Pipeline Manager for each BRM database schema.**

  Each instance of Pipeline Manager has a DAT_Listener module that retrieves events for that instance. You specify the events for each instance in the DAT_Listener registry.

> **Note:**
>
> It is recommended that you install an instance of the Account Synchronization DM and have a corresponding instance of Pipeline Manager for every BRM database schema, but it is not required. How you configure multiple database schemas depends on your business needs and should be determined by your system and database administrators.

Figure 76-2 shows the data flow from the Account Synchronization DM to multiple instances of Pipeline Manager:

Figure 76-2 Data Flow from Account Synchronization DM to Pipeline Manager Instances



About the Payload Configuration File

The Account Synchronization DM is an EAI publisher. All EAI publishers in your BRM system use the same payload configuration file. This file defines how EAI applications accumulate and format published information. The name and location of this file is specified in the **infranet.eai.configFile** entry in the EAI properties file (*BRM_home***/sys/eai_js/Infranet.properties**).

The Account Synchronization DM includes a payload configuration file (**payloadconfig_ifw_sync.xml**). This file defines the business events that the Account Synchronization DM uses to update data in Pipeline Manager.

If your BRM system *does not already have* an EAI-based publisher, the Account Synchronization installation program sets the **infranet.eai.configFile** entry to point to the Account Synchronization payload configuration file.

If your BRM system *already has* an EAI-based publisher installed, the Account Synchronization installation program automatically merges the Account Synchronization payload configuration file with the existing payload configuration file. The merged file is named

payloadconfig_MergedWithIfw_sync.xml, and the **infranet.eai.configFile** entry is updated accordingly. The names and contents of the original files used to create the merged file are not changed by the Account Synchronization installation program.

> **Note:**
>
> In rare cases, it may not be possible to use the merged payload configuration file. This can happen when entries in the two original payload configuration files have conflicting definitions. You must check for conflicts between the two original files *before* running the Account Synchronization DM.

About Controlling the Business Event Backlog

When you use Pipeline Manager for batch rating, the Account Synchronization database queue can become backlogged with business events waiting to be processed. This can delay account synchronization and slow Pipeline Manager performance, causing a greater number of CDRs to be suspended.

To control the number of events waiting to be processed, you can configure DAT_Listener to interleave CDR and business event processing. For information about how interleaved processing works, see "About Interleaved Processing".

Why Event Backlog Occurs

Event backlog can occur for the following reasons:

- When Pipeline Manager modules update account data, they lock the accounts they are updating in each transaction. When more than one transaction must access the same account, the transactions must wait until the account is unlocked. If the account stays locked for a relatively long time, business events can become backlogged while they wait for processing.
- When business events arrive that change account information during CDR processing, Pipeline Manager must update its memory before continuing to rate events in the CDR. This requires accessing the database, which adds to the number of times the accounts are locked. With concurrent business event and CDR processing, there could be multiple updates required, further reducing overall Pipeline Manager performance.
- When an account is currently being billed, Pipeline Manager must receive a business event notifying it that billing is complete before it can process new event data records (EDRs) for the account. If a billing business event is waiting to be processed behind a backlog of other events, there might be many new EDRs for the account that require suspending until the billing event is processed. Each time a suspended EDR is recycled, it creates a new business event that is added to the account synchronization queue.

About Interleaved Processing

You can configure Pipeline Manager to process CDRs and business events exclusively rather than concurrently. With *interleaved processing*, business event and CDR processing are interleaved so that multiple business events can be processed before rating the CDRs, and multiple CDRs can be rated before updating Pipeline Manager memory. This balances the event backlog and improves processing time by decreasing the number of times Pipeline Manager must access the database and lock accounts. Exclusive processing reduces the

likelihood that you will need to recycle events due to delayed synchronization or to stop Pipeline Manager to catch up on business event processing.

DAT_Listener controls whether business events or CDRs are processed based on the following criteria:

- How many business events are waiting to be processed in the Account Synchronization queue.
- The time that Pipeline Manager has been currently processing either business events or CDRs.

You can use the first or both of these criteria, depending on how much control you want over the interleaving process. See "About Setting Processing Thresholds".

You specify the thresholds for event processing in the DAT_Listener module registry.

Assuming both the number of events and the processing time are configured, DAT_Listener performs the following actions to control interleaved processing:

1. Periodically checks the number of business events waiting to be processed in the Account Synchronization queue and tracks the amount of time that Pipeline Manager has been currently processing either business events or CDRs.
2. Stops pipeline CDR processing when the number of business events in the queue reaches the maximum threshold or when the maximum CDR processing time is reached, whichever comes first.

   DAT_Listener stops Pipeline Manager by sending a stop command to the pipeline controller. This command suspends CDR processing but allows business event processing to proceed.
3. Dequeues business events and sends them to pipelines for processing.
4. Stops dequeuing business events when the number of business events in the queue reaches the minimum threshold or when the maximum business event processing time is reached, whichever comes first.
5. Restarts Pipeline Manager CDR processing by sending a start command to the pipeline controller.

The Account Synchronization DM continues to send business events to the queue while Pipeline Manager processes CDRs.

When tracking the number of business events in the queue, DAT_Listener counts only the business events that have a state of READY. All other business events are not considered.

About Setting Processing Thresholds

For basic process interleaving, you set the minimum and maximum business event thresholds. For more control over CDR and business event interleaving, you can also set the maximum CDR and business event processing times.

For example, if you typically process fewer events in the middle of the night, it might take longer to reach the specified maximum number of business events in the queue. Account synchronization will be delayed during this time and you might have a higher incident of CDRs that need suspending or rerating. In this case, you limit the amount of time spent on CDR processing. When this time threshold is reached, DAT_Listener switches to business event processing even if the number of events in the queue has not reached the maximum.

You might want to limit business event processing time if there are certain times when many business events accumulate in the queue (for example, when you shut down Pipeline Manager

for an extended period). When you stop and restart Pipeline Manager, rather than waiting until the minimum business event threshold is reached, Pipeline Manager switches to CDR processing when the maximum business event process time is reached.

> **Note:**
>
> DAT_Listener does not check whether there are CDRs waiting to be processed. It checks only for business events and processing times. Therefore, if there are no CDRs to process, DAT_Listener waits until either the CDR processing time threshold or the maximum number of business events is reached before switching to business event processing. If this wait period is too long, you can update the DAT_Listener registry to switch from CDR to business event processing.

77

Installing and Configuring Account Synchronization

This chapter describes how to install and configure Oracle Communications Billing and Revenue Management (BRM) Account Synchronization Manager. Anyone who installs, configures, or administers Account Synchronization Manager should read this document.

About Installing Account Synchronization Manager

Before installing Account Synchronization Manager, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

Installing and Configuring Account Synchronization

To install Account Synchronization:

1. Configure your database machines for advanced queuing.

   See "Configuring Database Machines for Advanced Queuing".
2. Install Account Synchronization Manager.

   See "Installing Account Synchronization Manager".
3. Create additional Account Synchronization queues.

   See "Creating Additional Account Synchronization Queues".
4. If your system contains queues in multiple database schemas, grant execute permission for **acct_sync**.

   See "Granting Execute Permission for acct_sync".
5. Configure the EAI payload for Account Synchronization.

   See "Configuring the EAI Payload for Account Synchronization".
6. Enable event notification.

   See "Configuring Event Notification for Account Synchronization".
7. Configure Account Synchronization.

   See "Configuring Account Synchronization".
8. Configure the DAT_Listener module.

   See "Configuring the DAT_Listener Module".
9. Set up service-level bill items.

   See "Setting Up Service-Level Bill Items".
10. Enable BRM to audit GSM (Global System for Mobile Communication) objects.

    See "Turning On Object Auditing".
11. Configure Account Synchronization for multiple database schemas.

See "Configuring Account Synchronization for Multiple Database Schemas".

Configuring Database Machines for Advanced Queuing

Before you install Account Synchronization Manager, you must configure all queuing database machines for advanced queuing.

Perform the following on each queuing database machine in your system:

1. Open the *Oracle_home*/**dbs/init***SID*.**ora** file in a text editor, where *Oracle_home* is the directory in which you installed the Oracle Database.
2. Set the **compatible** parameter to your Oracle database version.

   > **Note:**
   >
   > If you use an entry lower than **10.0**, your BRM and Pipeline Manager systems fail.

3. Specify one queue monitor process (QMn) by setting the **aq_tm_process** parameter to **1**.

   The queue monitor process removes from the queue any processed events that are over an hour old.
4. Save and close the file.
5. Using SQL*Plus, log in to your database as the SYS user and grant advanced queuing privileges to each schema user in your system:

   ```
   % sqlplus sys/password@databaseAlias

   SQL> grant execute on dbms_aq to schema_user_name;

   Grant succeeded.

   SQL> grant execute on dbms_aqadm to schema_user_name;

   Grant succeeded.

   SQL> grant select on sys.gv_$aq to schema_user_name;

   Grant succeeded.

   SQL> grant execute on dbms_lock to schema_user_name;

   Grant succeeded.
   ```

   where *schema_user_name* is the user name for the schema to which you are granting privileges.
6. Initialize the database instance with your changes by stopping and restarting the Oracle database.

Installing Account Synchronization Manager

Before installing Account Synchronization Manager, you must install BRM and Multidatabase Manager (Multischema systems only).

> **Note:**
>
> If you have already installed the product, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

To install Account Synchronization Manager, see "Installing Individual BRM Components" in *BRM Installation Guide* for instructions.

Creating Additional Account Synchronization Queues

The Account Synchronization Manager installer automatically creates a default queue in a specified database schema. If your system requires multiple queues, create additional queues by manually running the **pin_ifw_sync_oracle** utility.

> **Note:**
>
> To avoid system errors, do not run Pipeline Manager while running the **pin_ifw_sync_oracle** utility.

To create an additional database queue:

1. Verify the default queue storage and retention time settings in the **create_ifw_sync_queue.conf** file.

   For more information, see:

   - Configuring the Queue Location
   - Configuring How Often Processed Events Are Removed from the Queue

2. Enter the following command, which logs you in as the **pin** user:

```
su - pin
```

> **Note:**
>
> To create queues, the **pin** user must have Oracle Advanced Queuing (AQ) privileges.

3. Run the following command, which creates a new queue:

```
pin_ifw_sync_oracle.pl create -q queue_name -t queue_table -l /@database_alias
```

   where:

   - *queue_name* is the name of the new queue. If you omit this parameter the utility creates a database queue named IFW_SYNC_QUEUE.

> **Note:**
>
> In multischema systems, all queues in a schema must have unique names, but the same queue name can be used across multiple schemas.

- *queue_table* is the name of the database table that contains the *queue_name*. If you omit this parameter, the utility creates a queue table named IFW_SYNC.
- *database_alias* is the BRM database alias of the database to which you are adding a queue. If you omit this parameter the utility will prompt you for this information.

4. Run the following command, which verifies that the queue was created and functions properly by attempting to enqueue and dequeue 20 test events:

```
pin_ifw_sync_oracle.pl test -q queue_name -l /@database_alias
```

The utility enqueues and dequeues 20 test events.

> **Note:**
>
> You only need to test a queue after it has first been created.

5. Run the following command, which generates a summary report that displays the state of each event in the queue:

```
pin_ifw_sync_oracle.pl report -r summary -q queue_name -l /@database_alias
```

If the tests were successful, the report lists the 20 events with a processed state:

```
Tue Feb 04                                                    page 1
                        Event Summary Report
Evt. Stat  Event Name                       Event Count
---------  -------------------------------  -----------
PROCESSED  LongTestEvent                              10
           ShortTestEvent                             10
*********                                   -----------
sum                                                   20
```

If unsuccessful, you must drop and then recreate the queue and queue table. For information, see "Dropping the Queue and Queue Tables".

Creating an Acknowledgment Queue for Account Migration Manager

In a system containing Account Migration Manager, an acknowledgment queue pushes AMM-related events. Multiple pipelines, associated with different schemas, can connect to this one acknowledgment queue and send acknowledgment events. If events are enqueued into an acknowledgment queue that resides in a different schema, you must specify the schema name in the **create_ifw_sync_queue.conf** file.

To modify the default schema name in the **$primary_schema** parameter:

1. Open the *BRM_home*/**apps/pin_ifw_sync/create_ifw_sync_queue.conf** file in a text editor.
2. Add or modify the following entry:

```
$primary_schema = "schema_name";
```

where *schema_name* is the name of the schema that contains the acknowledgment queue into which events are enqueued. Use **pin** if the acknowledgment queue resides in the local schema.

> **Note:**
>
> For example, if the acknowledgment queue is configured in schema **pin01** and the events from schema **pin02** have to be enqueued into that acknowledgment queue:
>
> ```
> $primary_schema = "pin01"
> ```

3. Save and close the file.
4. To enable access to the schema where the acknowledgment queue resides:
   a. Create a new *file_name***.sql** file.
   b. Add the following commands to *file_name***.sql**:

```
set serveroutput on;
BEGIN
EXECUTE IMMEDIATE 'GRANT EXECUTE ON PIN_EVENT_TY TO schema_name ;
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE(
privilege => 'ALL',
queue_name => 'queue_name',
grantee => 'schema_name',
grant_option => FALSE);
EXECUTE IMMEDIATE 'GRANT SELECT ON AQ$queue_table TO schema_name';
EXECUTE IMMEDIATE 'GRANT UPDATE ON AQ$queue_table TO schema_name';
END;

----------------------------------------------------------------------
```

where:

queue_name is the acknowledgment queue name.

schema_name is the schema from which events are enqueued to the acknowledgment queue.

queue_table is the database table that contains the acknowledgment queue.

> **Note:**
>
> Execute the GRANT statements for each schema from which events are enqueued to the acknowledgment queue.

For example:

```
BEGIN
EXECUTE IMMEDIATE 'GRANT EXECUTE ON PIN_EVENT_TY TO PIN02 ;
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE(
privilege => 'ALL',
queue_name => 'IFW_SYNC_QUEUE_AMT',
grantee => 'PIN02',
grant_option => FALSE);
EXECUTE IMMEDIATE 'GRANT SELECT ON AQ$IFW_SYNC_QUEUE_AMT_T TO PIN02';
```

```
    EXECUTE IMMEDIATE 'GRANT UPDATE ON AQ$IFW_SYNC_QUEUE_AMT_T TO PIN02';
    END;
```

5. Open SQL *Plus by entering the following command:

   sqlplus pin/*password@ORACLE_SID*

   where *ORACLE_SID* is the BRM database alias.

6. From the schema where the acknowledgment queue is configured, run the following command:

   @file_name.**sql**

7. Run the **pin_ifw_sync_oracle** utility. See "Creating Additional Account Synchronization Queues" for more information.

Granting Execute Permission for acct_sync

If your system contains queues in multiple database schemas, you must grant execute permission for the **acct_sync** package from each queuing database schema to every other queuing database schema. This enables **dm_ifw_sync** to forward events to queues residing on other database schemas.

> **Note:**
>
> If you do not have a multischema system, skip this section and go to "Configuring the EAI Payload for Account Synchronization".

To grant execute permission for **acct_sync**:

1. Log in to a target database schema:

   ```
   % sqlplus target_schema_user_name@database_alias
   Enter password: password
   ```

   where:

   - *target_schema_user_name* is the user name for a schema *to* which events are forwarded from a source schema.
   - *password* is the password for the target schema user name.
   - *database_alias* is the BRM database alias of the target schema.

2. Grant execute privileges to the source schema:

   ```
   SQL> grant execute on acct_sync to source_schema_user_name
   ```

   where *source_schema_user_name* is the user name for the schema *from* which events are forwarded to the target.

For example, if **dm_ifw_sync** connects to the **pin1** database schema, enable it to send an event to the **pin2** schema as follows:

1. Log in to the **pin2** schema:

   ```
   % sqlplus pin2@database_alias
   Enter password: password
   ```

   where:

- *password* is the password for the **pin2** schema user name.
- *database_alias* is the BRM database alias of the **pin2** schema.

2. Grant execute privileges to the **pin1** schema:

```
SQL> grant execute on acct_sync to pin1
```

Configuring the EAI Payload for Account Synchronization

You must modify your BRM system's payload configuration file in the following situations:

- If you have another EAI-based publisher, you must do the following:
  - Check for conflicts in the EAI payload configuration files.

    See "Checking for Conflicts in EAI Payload Configuration Files".
  - Specify the default configuration file.

    See "Specifying the Default Payload Configuration File".
- If your Account Synchronization DM database number is not the default (0.0.9.9). See "Specifying the Account Synchronization DM Database Number ".
- If you uninstall the Account Synchronization DM. See "Revising the Payload Configuration File When Uninstalling Account Synchronization Manager".

Checking for Conflicts in EAI Payload Configuration Files

If your BRM system already has an EAI publisher, the Account Synchronization Manager installation program merges this file with the existing payload configuration file.

> **Note:**
>
> If you do not have an existing EAI publisher application, skip this section and go to "Specifying the Account Synchronization DM Database Number ".

In rare cases, conflicts can occur when the Account Synchronization Manager payload configuration file is merged with an existing payload configuration file. This can happen when entries in the two original configuration files have conflicting definitions.

> **Note:**
>
> You must determine whether merge conflicts exist *before* running Account Synchronization Manager.

Compare the Account Synchronization Manager payload configuration file (*BRM_home***/sys/eai_js/payloadconfig_ifw_sync.xml**) with the existing payload configuration file (referenced in *BRM_home***/sys/eai_js/Infranet.properties**). If any of the following conditions exist, it is not possible to run both EAI publishers in the same BRM system:

- Two different business event definitions specify the same **StartEvent**.
- The same business event definitions have different values for one or more of these attributes: **StartEvent**, **EndEvent**, or **Source**.

- The same business event or element definitions have different search criteria.
- The same element definitions have different values for one or more of the following attributes: **Source**, **PinFld**, **DataFrom**, **UseOnlyElement**, or **Tag**.
- The same element definitions have different **OnEvent** values.

Specifying the Default Payload Configuration File

If your BRM system already has an EAI publisher, you must make the **payloadconfig_MergedWithIfw_sync.xml** file your system's default payload configuration file.

> **Note:**
>
> If you do not have an existing EAI publisher application, skip this section and go to "Specifying the Account Synchronization DM Database Number ".

To specify the default payload configuration file, do one of the following:

- Change the EAI **configFile** entry so that it points to the merged payload configuration file:
  1. Open the *BRM_home***/sys/eai_js/Infranet.properties** file in a text editor.
  2. Set the **infranet.eai.configFile** entry to point to the *BRM_home***/sys/eai_js/payloadconfig_MergedWithIfw_sync.xml** file:

     ```
     infranet.eai.configFile=./payloadconfig_MergedWithIfw_sync.xml
     ```

  3. Save and close the file.
- Rename the configuration file and the **configFile** entry to match each other:
  1. Change the name of the file specified in the **infranet.eai.configFile** entry.

     For example, change **payloadconfig.xml** to **payloadconfig_eai.xml.**
  2. Change the name of the *BRM_home***/sys/eai_js/payloadconfig_MergedWithIfw_sync.xml** file to match the file name specified in the **infranet.eai.configFile** entry.

Specifying the Account Synchronization DM Database Number

The default BRM database number for your Account Synchronization DM is 0.0.9.9. If you change your database number, you must change the value of the **DB** attribute in the Account Synchronization DM publisher definition.

> **Note:**
>
> If you did not change the default database number for your Account Synchronization DM, go to "Configuring Event Notification for Account Synchronization".

To specify the Account Synchronization DM database number:

1. Open the *BRM_home***/sys/eai_js/payloadconfig_ifw_sync.xml** file, or open the merged file if you merged payload configuration files, in a text editor.

2. Find the **<PublisherDefs>** section.
3. In the **Publisher DB** entry, enter the correct BRM database number.

   For example, if your Account Synchronization DM database number is 0.0.9.5, change this entry:

   ```
   <PublisherDefs>
       <Publisher DB="0.0.9.9" Format="FLIST">
   ```

   to this:

   ```
   <PublisherDefs>
       <Publisher DB="0.0.9.5" Format="FLIST">
   ```

4. Save and close the file.

Revising the Payload Configuration File When Uninstalling Account Synchronization Manager

To remove Account Synchronization elements from your system's EAI payload configuration file after uninstalling Account Synchronization Manager:

1. Open the *BRM_home***/sys/eai_js/payloadconfig_ifw_sync.xml** file, or open the merged file if you merged payload configuration files, in a text editor.
2. Find the <**PublisherDefs**> section.
3. Remove the following publisher definition:

   ```
   <Publisher DB="database_number" Format="FLIST">
   ```

4. Save and close the file.
5. Stop and restart the Payload Generator External Module (the EAI Java server) by entering the following command from the *BRM_home***/bin** directory:

   ```
   pin_ctl bounce eai_js
   ```

Configuring Event Notification for Account Synchronization

When a BRM event that is included in a business event defined in the Account Synchronization Manager payload configuration file occurs, the EAI framework uses event notification to call the opcode that caches the BRM event in the Payload Generator.

To configure the event notification feature:

1. If your system has multiple configuration files for event notification, merge them.
2. Ensure that the merged file includes the entire event notification list in the *BRM_home***/sys/data/config/pin_notify_ifw_sync** file.
3. (Optional) If you defined new business events for Account Synchronization, you must edit your final event notification list to include all the BRM events in the new business events.
4. Load your final event notification list into the BRM database.

Configuring Account Synchronization

To configure the Account Synchronization DM to connect to the BRM database and the database queue:

1. Connect the Account Synchronization Manager's EAI framework to BRM by editing the CM configuration file (**pin.conf**).

   See "Configuring the CM for Account Synchronization".

2. Map BRM business events to the appropriate database queue by editing the **ifw_sync_queuenames** file.

   See "Mapping Business Events to Database Queues".

3. Connect the Account Synchronization DM to the database queue by editing the Account Synchronization DM configuration file (**pin.conf**).

   See "Configuring the Account Synchronization DM".

Configuring the CM for Account Synchronization

You must modify the CM **pin.conf** file to enable the EAI framework to notify Account Synchronization Manager when specific events occur. You should also verify that the pointer to the Account Synchronization DM specifies the correct database and port numbers.

> **Note:**
>
> If you use a multischema system and set up more than one CM, you must edit the configuration file for each CM.

To configure the CM for account synchronization:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **enable_publish** entry to **1**:

   ```
   - fm_publish enable_publish 1
   ```

3. Verify that the **dm_pointer** entry for **dm_ifw_sync** specifies the correct database number and port:

   ```
   - cm dm_pointer 0.0.9.9 ip host_name/ip_address 11999
   ```

   where:

   - **0.0.9.9** is the default Account Synchronization DM database number.
   - *host_name/ip_address* is the host name or IP address of the computer on which the Account Synchronization DM runs.
   - **11999** is the default Account Synchronization DM port number.

   > **Note:**
   >
   > If you change the location of the Account Synchronization DM, you must modify this entry.

> **Note:**
>
> The Account Synchronization DM *database number* must match the number in the following entries:
>
> – The **dm_db_no** entry in the Account Synchronization DM configuration file (*BRM_home***/sys/dm_ifw_sync/pin.conf**).
>
> – The **DB** entry of the Account Synchronization Publisher definition in the payload configuration file (*BRM_home***/sys/eai_js/payloadconfig_ifw_sync.xml**).
>
> The Account Synchronization DM *port number* must match the number in the **dm_port** entry in the Account Synchronization DM configuration file (*BRM_home***/sys/dm_ifw_sync/pin.conf**).

4. Save and close the file.
5. Stop and restart the CM:
   - From the *BRM_home***/bin** directory, enter this command:

     ```
     pin_ctl bounce cm
     ```

Mapping Business Events to Database Queues

You configure which events the Account Synchronization DM sends to each database queue by editing the **ifw_sync_queuenames** file. This file must specify the names of all the queues in your system and which events to send to each queue.

> **Note:**
>
> - Only business events that are defined in the **payloadconfig** file can be sent to a database queue.
> - Account synchronization does not send events to queues until you edit the **ifw_sync_queuenames** file.

To map business events to database queues:

1. Open the *BRM_home***/sys/dm_ifw_sync/ifw_sync_queuenames** file in a text editor.
2. Add an entry for each queue in your system by using the following syntax:

   ```
   schema_name.queue_name
   {
     criteria
   }
   ```

   where:

   - *schema_name* is the name of the database schema containing the queue.

     In multischema systems, each instance of the Account Synchronization DM connects to only one database queue in only one schema. That schema is the DM's local schema. In some cases, the Account Synchronization DM receives an event that belongs to a queue in a different schema (for example, when you move an account

from one schema to another). Oracle AQ uses the queue name entries in the **ifw_sync_queuenames** file to send the events to the appropriate queue:

  - If a queue is on the local database schema, do not include *schema_name* in the entry.
  - If a queue is on a remote database schema, prepend the queue name with the name of the database schema to which it belongs.

- *queue_name* is the name you assigned the queue when you created it. Each queue name must match the queue name in the corresponding DAT_Listener registry.
- *criteria* specifies which events to send to the queue. You can configure the Account Synchronization DM to send all business events, only events from a specific database schema, or only specific event types. Table 77-1 shows the syntax for each criteria:

Table 77-1 Syntax to Add Events to Queues

To Send This	Use This Syntax	Example
All business events	**ALL**	**ALL**
Only business events from a specific database schema	**0.0.0.***x*	**0.0.0.1**
Only specific event types	*eventName*	CustCreate ProductCancel
Excluding specific event types	!*eventName*	!CustCreate !ProductCancel
A specific event from a specific database	0.0.0.x AND *eventName*	0.0.0.1 AND ProductCancel
Excluding a specific event from a specific database	0.0.0.x AND !*eventName*	0.0.0.1 AND !ProductCancel

For example, to send all business events from BRM database schema 0.0.0.1 to the local IFW_SYNC_QUEUE_1 queue and only CustCreate events to the IFW_SYNC_QUEUE_2 queue, which resides in another database schema, use the following syntax:

```
IFW_SYNC_QUEUE_1
{
  0.0.0.1
}

schema_name.IFW_SYNC_QUEUE_2
{
  CustCreate
}
```

3. Save and close the file.

Mapping Events in Systems Containing Multiple ifw_sync_queuenames Files

In a multischema system, each BRM installation machine on which an instance of the Account Synchronization DM is installed contains an **ifw_sync_queuenames** file for that DM.

The **ifw_sync_queuenames** file of the primary Account Synchronization DM must include one or more queue entries for every database schema in your system.

For example, if you use three database schemas, the entries in the **ifw_sync_queuenames** file of the primary Account Synchronization DM, which is connected to schema 0.0.0.1, look like this:

```
IFW_SYNC_ROUTER_QUEUE
{
 CustCreate
 ServiceLogin
 CustDelete
}

IFW_SYNC_QUEUE_1      # queue on local database schema
{
  0.0.0.1
}
0.0.0.2.IFW_SYNC_QUEUE_2      # queue on remote database schema
{
  0.0.0.2
}
0.0.0.3.IFW_SYNC_QUEUE_3      # queue on remote database schema
{
  0.0.0.3
}
```

> **Note:**
>
> In the preceding example, the queue that connects to the Pipeline Manager instance containing the multischema account router (IFW_SYNC_ROUTER_QUEUE) is on the same database schema as the primary queue (IFW_SYNC_QUEUE_1).
>
> The multischema account router is interested in only **CustCreate**, **ServiceLogin**, and **CustDelete** events. However, you can alternatively specify **ALL** to have all events sent to the router.

Any **ifw_sync_queuenames** file on a secondary installation machine needs to include entries for only local queues. For example, the entry in the **ifw_sync_queuenames** file of a secondary Account Synchronization DM connected to schema 0.0.0.2 looks like this:

```
IFW_SYNC_QUEUE_2      # queue on local database schema
{
  0.0.0.2
}
```

Configuring the Account Synchronization DM

During installation, the Account Synchronization DM installer generates a **pin.conf** configuration file that specifies how to connect to your BRM database and database queue and that contains other configuration settings. The installer populates the connection entries with values from your **pin_setup.values** file and provides default information for the other configuration entries. Before you start the Account Synchronization DM, verify that the file contains accurate information for connecting to your BRM database and database queue.

In multischema systems, each BRM database schema might have its own Account Synchronization DM. Verify that each Account Synchronization DM **pin.conf** file contains accurate information for its associated database schema and queue.

To configure the Account Synchronization DM:

> **Note:**
>
> Do this for each instance of Account Synchronization DM.

1. Open the (*BRM_home*/**sys/dm_ifw_sync/pin.conf**) file in a text file.
2. Verify that the **plugin_name** entry specifies the path and file name of the correct shared library file:

   ```
   -dm plugin_name path/name
   ```

   where *name* is **libplugin_ifw_sync***database_version***.so**

   database_version is the version number of your Oracle database.
3. Verify that the **queue_map_file** entry specifies the path and file that maps the database queues to the events they are to receive.

   For example:

   ```
   - dm_ifw_sync queue_map_file ./ifw_sync_queuenames
   ```

   For more information, see "Mapping Business Events to Database Queues".
4. Verify that the **sm_database** entry specifies the alias name of the queuing database to which the Account Synchronization DM connects.

   The value of this entry should be the TNSNAMES.ORA alias, which can be found in the *Oracle_home*/**network/admin/tnsnames.ora** file:

   ```
   - dm_ifw_sync sm_database alias_name
   ```
5. Verify that the **sm_id** entry specifies the database schema user name that the Account Synchronization DM uses to log in to the queuing database schema:

   ```
   - dm_ifw_sync sm_id user_name
   ```
6. (Optional) You can also edit the entries in Table 77-2:

   Table 77-2 Entries That Are Editable

Entry Name	Description
connect_retries	Specifies the number of times that the Account Synchronization DM attempts to connect to the database queue.
retry_interval	Specifies the length of time, in seconds, that the Account Synchronization DM waits before attempting to reconnect to the database queue.

   For more information, see "Configuring Account Synchronization DM Database Connection Attempts".

   For other entries you can edit, see the comments in the **pin.conf** file.
7. Save and close the file.

Configuring the DAT_Listener Module

Each instance of Pipeline Manager is connected to one database queue. You must configure each instance's DAT_Listener module to connect to its corresponding database queue by modifying the Pipeline Manager registry.

Configuring the Registry

Perform the following tasks for each instance of Pipeline Manager:

1. Open the registry file.
2. Add the following entries to the Listener section of the registry file:

> **Note:**
>
> The Listener section must be listed after the pipeline and BRM database connection sections. Otherwise, Pipeline Manager fails to start.

```
Listener
{
  ModuleName = DAT_Listener
  Module
  {
    InfranetConnection =              #mandatory
    QueueName =                       #mandatory
    QueueLibrary =                    #mandatory
    ConnectRetries =                  #optional
    RetryInterval =                   #optional
    LogEvents =                       #optional
  }
}
```

where:

- **InfranetConnection** points to the section of the registry file that specifies how the Account Synchronization DM connects to the database queue.

  For example, **ifw.DataPool.LoginAccountSync**. This entry is mandatory.

- **QueueName** specifies the queue name.

  For example, IFW_SYNC_QUEUE. This entry is mandatory.

- **QueueLibrary** specifies if the Listener should use Oracle queue libraries. Enter **OracleQueue**.

- **ConnectRetries** specifies the number of times the DAT_Listener module retries to connect to the database queue.

  This entry is optional. The default is **0**.

- **RetryInterval** specifies the time, in seconds, between reconnection attempts.

  This entry is optional. The default is **5**.

- **LogEvents** specifies whether the entire contents of each business event is copied to a log file.

  This entry is optional. The default is **FALSE**.

3. Save and close the file.

Configuring Interleaved Processing

You can control how many events are waiting to be processed simultaneously by configuring DAT_Listener to interleave call detail record (CDR) and business event processing.

> **Note:**
>
> The following entries are optional. If not present, Pipeline Manager processes CDRs and business events concurrently.

To interleave processing between CDR and business events:

1. Open the registry file.
2. Add the following entries to the Listener section of the registry file:

```
Listener
{
   ModuleName = DAT_Listener
   Module
   {
    #Add the following lines for interleaved processing:
    InterleavingReqd = true
    MaxNumEvents =
    MinNumEvents =
    CheckInterval =
    EnableInterLeavingStatistics =
    ProcessAllEvents =
    MaxEventProcessTime =
    MaxCDRProcessTime =
   }
}
```

where:

- **InterleavingReqd** specifies whether interleaved processing is enabled.

  The default is **False**. When set to **False** or not specified, interleaved processing is not performed; CDRs and events are processed simultaneously.

- **CheckInterval** specifies (in seconds) how frequently DAT_Listener checks the number of events waiting in the queue (specified by **MaxNumEvents** and **MinNumEvents**) and the amount of processing time that has passed (specified by **MaxCDRProcessTime** and **MaxEventProcessTime**).

  The default is **60**. If this entry is not present, the default interval is used.

> **Note:**
>
> This entry takes precedence over **MaxNumEvents, MinNumEvents, MaxEventProcessTime,** and **MaxCDRProcessTime**. If **CheckInterval** is set too high, DAT_Listener will not switch between CDR and event processing even if the thresholds have been reached. For example, if **MaxEventProcessTime** is set as 3600 seconds and **CheckInterval** is set to 7200 seconds, events are processed for 7200 seconds before the processing time is checked.

- **MaxNumEvents** specifies the maximum number of business events allowed in the queue.

  The default is **900**. When the number of events in the queue reaches or exceeds this amount, DAT_Listener stops CDR processing and starts business event processing. When this entry is specified, **MinNumEvents** must also be specified.

- **MinNumEvents** specifies the minimum number of business events allowed in the queue.

  The default is **300**. When the number of events in the queue reaches or drops below this amount, DAT_Listener stops business event processing and starts CDR processing. When this entry is specified, **MaxNumEvents** must also be specified.

- **EnableInterLeavingStatistics** specifies whether to log only interleaving statistical data.

  The default is **False**. By default, DAT_Listener logs all processing messages in the pipeline processing log (**process.log**). If this entry is set to **True**, only statistical data related to the rate of interleaved processing is logged. You can use this entry to monitor your event processing performance.

- **ProcessAllEvents** specifies whether to process all events in the queue when Pipeline Manager is started.

  A value of **True** processes all events in the queue before activating interleaved processing. A value of **False** activates interleaved processing at startup, and events are processed according to the interleaving settings.

  If set to **True** at startup, after processing all events, this entry is reset to **False**. To use this feature, you must reset this entry to **True** each time you restart Pipeline Manager.

- **MaxEventProcessTime** specifies the maximum number of seconds that business events are processed.

  The default is **60**. When Pipeline Manager has been processing business events for this amount of time, DAT_Listener stops business event processing and starts CDR processing regardless of how many business events are in the queue. When this entry is specified, **MaxNumEvents**, **MinNumEvents**, and **MaxCDRProcessTime** must also be specified.

- **MaxCDRProcessTime** specifies the maximum number of seconds that CDRs are processed.

  The default is **300**. When Pipeline Manager has been processing CDRs for this amount of time, DAT_Listener stops CDR processing and starts business event processing regardless of how many CDRs are in the queue. When this entry is specified, **MaxNumEvents**, **MinNumEvents**, and **MaxEventProcessTime** must also be specified.

> **Note:**
>
> The default values for interleaved processing are also the minimum required values. If you specify a value less than the default for any entry, that value is ignored and the default value is used.
>
> Some entries can be used in a semaphore to update the processing thresholds. If these semaphores contain an error, the pipeline is deactivated and you must reactivate it.

3. Save and close the file.

Switching to Business Event Processing When No CDRs Are Waiting

When there are no CDRs to process, DAT_Listener waits until either the CDR processing time threshold is reached or the maximum number of business events is reached before switching to business event processing. If this wait period is too long, you can shorten the CDR processing time by using a semaphore to update the DAT_Listener registry. The new semaphore value becomes effective the next time DAT_Listener checks for the number of events and elapsed processing time. This is determined by the **CheckInterval** registry entry. For example, if **CheckInterval** is set to 180 seconds, the semaphore update becomes effective after 180 seconds have elapsed.

Use one of the following semaphores to shorten the CDR processing time so that business events are processed sooner:

- Shorten the CDR processing time by changing the **MaxCDRProcessTime** value:

  ```
  ifw.DataPool.Listener.Module.MaxCDRProcessTime=new_value
  ```

- Reduce the maximum number of business events by changing the **MaxNumEvents** value:

  ```
  ifw.DataPool.Listener.Module.MaxNumEvents=new_value
  ```

  If the number of business events waiting in the queue exceeds the new value, DAT_Listener immediately switches to business event processing when it checks the number of events.

Setting Up Service-Level Bill Items

To enable Pipeline Manager to choose the correct bill item for an event, you must configure BRM to pre-create service-level items.

To set up service-level bill items, run the **load_config_item** utilities to load the contents of the **config_item_tags** and **config_item_types** files into the BRM database.

Turning On Object Auditing

After installing Account Synchronization Manager, you must enable BRM to audit GSM objects. Auditing objects creates a history of certain information, such as phone numbers and logins, so that BRM can track changes.

For a list of default objects, see "Required Objects to Be Audited".

The **object_auditing.pl** script invokes the **pin_history_on** utility to load the file containing the objects to audit and turn on object auditing. It then creates the audit table indexes in the BRM database.

Caution:

- When installing wireless managers such as Account Synchronization Manager or GSM Manager, you must install *all* managers *before* running this script. If you run this script before installing a wireless manager, your installation will fail.
- If you use a multischema system, do the following:
  1. Install all wireless managers.
  2. Install your multischema system.
  3. Run the **object_auditing.pl** script.

To enable additions of GSM objects, enter the following command:

```
perl object_auditing
```

Note:

The **object_auditing.pl** script requires a **pin.conf** configuration file. You can create one, or you can run the utility from a directory that contains a **pin.conf** file, such as *BRM_home***/sys/dm_ifw_sync**.

You can customize audit table indexes or the list of audited objects by modifying the input files before running the **object_auditing.pl** script:

- To customize the list of audited objects, edit the *BRM_home***/apps/integrate_sync/pin_history_on_input** file.
- To customize your audit table indexes, edit the *BRM_home***/sys/dd/data/create_indexes_audit_tables_oracle.source** file.

Required Objects to Be Audited

Pipeline Manager requires certain objects to be audited, which are included in the *BRM_home***/apps/integrate_sync/pin_history_on_input** file. Table 77-3 lists the default objects and their fields configured in the file:

Table 77-3 Objects Configured in pin_history_on_input

Object	Fields Audited by Default
/account	• PIN_FLD_BRAND_OBJ • PIN_FLD_ACCOUNT_NO • PIN_FLD_CURRENCY • PIN_FLD_RESIDENCE_FLAG • PIN_FLD_PRODUCTS.PIN_FLD_USAGE_END_T • PIN_FLD_PRODUCTS.PIN_FLD_USAGE_START_T

Table 77-3 (Cont.) Objects Configured in pin_history_on_input

Object	Fields Audited by Default
/profile/acct_extrating	• PIN_FLD_DATA_ARRAY.PIN_FLD_NAME • PIN_FLD_DATA_ARRAY.PIN_FLD_VALUE • PIN_FLD_DATA_ARRAY.PIN_FLD_VALID_FROM • PIN_FLD_DATA_ARRAY.PIN_FLD_VALID_TO
/profile/serv_extrating	• PIN_FLD_DATA_ARRAY.PIN_FLD_NAME • PIN_FLD_DATA_ARRAY.PIN_FLD_VALUE • PIN_FLD_DATA_ARRAY.PIN_FLD_VALID_FROM • PIN_FLD_DATA_ARRAY.PIN_FLD_VALID_TO
/service	• PIN_FLD_AAC_PROMO_CODE • PIN_FLD_AAC_VENDOR • PIN_FLD_ALIAS_LIST • PIN_FLD_ALIAS_LIST.PIN_FLD_NAME • PIN_FLD_LOGIN • PIN_FLD_STATUS • PIN_FLD_AAC_SOURCE
/uniqueness	• PIN_FLD_LOGIN • PIN_FLD_ALIAS_LIST.PIN_FLD_NAME • PIN_FLD_SERVICE_OBJ

Configuring Account Synchronization for Multiple Database Schemas

To synchronize accounts in BRM and Pipeline Manager in a multischema system:

1. Set up your multischema system and create multiple instances of Pipeline Manager for each database schema.
2. Configure each queuing database machine for advanced queuing.
3. Install the Account Synchronization DM on all systems that have the CM installed.

> **Note:**
>
> You do not need to perform all the configuration steps for secondary Account Synchronization installations. For example, you do not need to configure the EAI payload, enable event notification, set up service-level items, turn on object auditing, and validate charges. These updates are propagated to each database schema by Multidatabase Manager.

4. (Optional) Create additional database queues.

   See "Creating Additional Account Synchronization Queues".
5. Grant execute permission for **acct_sync** to your queuing database schemas so that they can forward events to queues in other schemas when necessary.

   See "Granting Execute Permission for acct_sync".
6. Map each instance of the DAT_Listener module to its corresponding database queue.

   See "Configuring the DAT_Listener Module".
7. Map the business events to each database queue in the **ifw_sync_queuenames** files.

See "Mapping Business Events to Database Queues".

8. Edit the configuration file (**pin.conf**) for each instance of the Account Synchronization DM.

   See "Configuring the Account Synchronization DM".

9. In the registry section for the DAT_AccountBatch module, add the **UseAsRouter** entry and set it to **True**, as in this example:

```
CustomerData
    {
    ModuleName = DAT_AccountBatch
    Module
        {
        IntegrateConnection = ifw.DataPool.Login
        InfranetConnection  = ifw.DataPool.LoginInfranet
        LogEvents           = True
        LoadLogins          = True
        Listener            = ifw.DataPool.Listener
        Connections         = 10
        Threads             = 10
        UseAsRouter         = True
        }
    }
```

10. In the registry section for the FCT_AccountRouter module, map each BRM database schema identifier to an output stream, as in this example:

> **Note:**
>
> You can use any names for the output streams.

```
AccountRouter
    {
    ModuleName = FCT_AccountRouter
    Module
        {
        Active            = True
        DataModule        = ifw.DataPool.CustomerData
        Streams
        {
        1 = PinOutputStream1
        2 = PinOutputStream2
        3 = PinOutputStream3
        4 = PinOutputStream4
        5 = PinOutputStream5
        }
      }
    }
```

11. Configure the registry of each Pipeline Manager instance to read accounts from its associated database schema.

    For example, Pipeline Manager 1 connects to database schema 1, Pipeline Manager 2 connects to database schema 2, and so on.

Starting and Stopping the Account Synchronization DM

To start the Account Synchronization DM, enter the following command at the prompt for Oracle AQ:

```
pin_ctl start dm_ifw_sync
```

To stop the Account Synchronization DM, enter the following command at the prompt for Oracle AQ:

```
pin_ctl stop dm_ifw_sync
```

Monitoring and Maintaining the Account Synchronization Queue

This section provides information and guidelines to help you manage your Account Synchronization queues.

The main administrative tasks for database queues are the following:

- Creating Additional Queues
- Generating Queue Reports
- Dropping the Queue and Queue Tables
- Configuring the Queue Location
- Configuring How Often Processed Events Are Removed from the Queue
- Configuring Account Synchronization DM Database Connection Attempts
- Disconnecting and Reconnecting the Account Synchronization DM to the Queue
- Disconnecting and Reconnecting DAT_Listener to the Queue

Creating Additional Queues

You must create additional queues in the following situations:

- You add a new instance of Pipeline Manager to your system.
- You add a BRM database schema to your system.

To create additional queues in an existing system:

1. Stop the Account Synchronization DM.

   See "Starting and Stopping the Account Synchronization DM".
2. If necessary, configure the Oracle database for Account Synchronization.

   See "Configuring Database Machines for Advanced Queuing".
3. If you are creating your queue in a new database schema, grant execute permission for its **acct_sync** package to your other queuing database schemas so they can forward events to the new database queue.

   See "Granting Execute Permission for acct_sync".
4. Create the new database queue by using **pin_ifw_sync_oracle**.

   See "Creating Additional Account Synchronization Queues".
5. Specify which events the Account Synchronization DM sends to the new queue.

See "Mapping Business Events to Database Queues".

6. Connect the new queue to its corresponding DAT_Listener module.

   See "Configuring the DAT_Listener Module".

7. Restart the Account Synchronization DM.

   See "Starting and Stopping the Account Synchronization DM".

Generating Queue Reports

You can monitor the events in your database queue by running the **pin_ifw_sync_oracle** utility.

> **Note:**
>
> To avoid system errors, do not run Pipeline Manager while running **pin_ifw_sync_oracle**.

pin_ifw_sync_oracle creates the following reports:

- A summary report that lists the number of events set to the READY and PROCESSED states.
- A detailed report that lists each event's ID, state, queuing time, and dequeuing time.

Generating Oracle AQ Reports

To run a summary report for Oracle AQ, enter the following command:

```
pin_ifw_sync_oracle.pl report -r summary [-q queue_name] -l /@database_alias
```

where:

- *queue_name* specifies the queue name.
- *database_alias* specifies the BRM database alias of the database schema.

To run a detailed report for Oracle AQ, enter the following command:

```
pin_ifw_sync_oracle.pl report -r detail [-q queue_name] -l /@database_alias
```

Dropping the Queue and Queue Tables

To drop the queue and its queue tables, enter the following command:

> **Note:**
>
> To avoid system errors, do not run Pipeline Manager while running **pin_ifw_sync_oracle**.

```
pin_ifw_sync_oracle.pl drop [-q queue_name]
```

where *queue_name* is dropped from the database.

If the database contains no other queues for Account Synchronization, the utility also removes the Account Synchronization package, which contains stored procedures for queuing, dequeuing, and purging events.

Configuring the Queue Location

By default, the **pin_ifw_sync_oracle** utility creates Oracle database queues in the tablespace you specified when you installed Account Synchronization Manager. To use a different tablespace, see one of the following:

- Specifying Default Storage Settings in the create_ifw_sync_queue.conf File
- Specifying Storage Settings by Using the pin_ifw_sync_oracle Utility

Specifying Default Storage Settings in the create_ifw_sync_queue.conf File

You can specify the default storage settings by using the **create_ifw_sync_queue.conf** file. All database queues that you create use the default settings unless you override them with the **pin_ifw_sync_oracle** utility.

To specify your default storage settings:

1. Open the *BRM_home*/**apps/pin_ifw_sync/create_ifw_sync_queue.conf** file in a text editor.
2. Specify the target tablespace and queue size by editing the **storage_clause** parameter:

> **Tip:**
>
> For production systems, create your queue in its own, separate tablespace to improve processing performance.

```
$storage_clause = "tablespace PIN00 initrans 5 storage (initial 200k next 200k
maxextents unlimited pctincrease 0 )";
```

3. Save and close the file.

Specifying Storage Settings by Using the pin_ifw_sync_oracle Utility

You can specify a queue's storage settings by using the **pin_ifw_sync_oracle** utility with the **-s** parameter. This option overrides the storage settings in the **create_ifw_sync_queue.conf** file.

> **Note:**
>
> To avoid system errors, do not run Pipeline Manager while running the **pin_ifw_sync_oracle** utility.

To specify storage settings by using the **pin_ifw_sync_oracle** utility, enter the following command:

```
su - pin
pin_ifw_sync_oracle.pl create [-q queue_name -t queue_table] -s storage_clause
```

where:

- *queue_name* specifies the queue name.
- *queue_table* specifies the queue table name.
- *storage_clause* specifies the queue's storage parameters.

> **Tip:**
>
> For production systems, create your queue in its own, separate tablespace to improve processing performance.

Configuring How Often Processed Events Are Removed from the Queue

The Oracle queue monitor process (QMn) removes from the queue any event that has been in the PROCESSED state for a specified amount of time. You specified a default retention time when you installed Account Synchronization Manager. To use a different retention time, see one of the following:

- Setting Default Retention Time in the create_ifw_sync_queue.conf File
- Setting Retention Times by Using the pin_ifw_sync_oracle Utility

Setting Default Retention Time in the create_ifw_sync_queue.conf File

You can set the default retention time in the **create_ifw_sync_queue.conf** file. All database queues that you create use this default setting unless you override it with the **pin_ifw_sync_oracle** utility.

To set the default retention time in the **create_ifw_sync_queue.conf** file:

1. Open the *BRM_home***/apps/pin_ifw_sync/create_ifw_sync_queue.conf** file in a text editor.
2. Set the **retention_time** parameter to the amount of time, in seconds, that you want to store processed events in the database queue:

   > **Note:**
   >
   > For production systems, set the retention time to **0** to optimize your processing performance.

   ```
   retention_time = retention_time;
   ```

3. Save and close the file.

Setting Retention Times by Using the pin_ifw_sync_oracle Utility

You can specify a queue's retention time by using the **pin_ifw_sync_oracle** utility with the **-r** parameter. This option overrides the retention time setting in the **create_ifw_sync_queue.conf** file.

> **Note:**
>
> To avoid system errors, do not run Pipeline Manager while running **pin_ifw_sync_oracle**.

To set the retention time by using the **pin_ifw_sync_oracle** utility:

```
su - pin
pin_ifw_sync_oracle.pl create [-q queue_name -t queue_table] -r retention_time
```

where:

- *queue_name* specifies the queue name.
- *queue_table* specifies the queue table name.
- *retention_time* specifies the queue's retention time, in seconds.

> **Tip:**
>
> For production systems, set the retention time to **0** to optimize your processing performance.

Configuring Account Synchronization DM Database Connection Attempts

You can configure how often the Account Synchronization DM attempts to connect to the database schema that contains the Account Synchronization queue.

To configure connection attempts:

1. Open the *BRM_home***/sys/dm_ifw_sync/pin.conf**) file in a text editor.
2. Specify the number of times the Account Synchronization DM should try to connect to the Oracle database server by editing the **connect_retries** entry.

   The default is **1**.

   ```
   -dm_ifw_sync connect_retries number_of_retries
   ```
3. Specify the interval, in seconds, between each reconnection attempt by editing the **retry_interval** entry.

   The default is **0**.

   ```
   -dm_ifw_sync retry_interval interval
   ```
4. Save and close the file.
5. Stop and restart the Account Synchronization DM.

   See "Starting and Stopping the Account Synchronization DM".

Disconnecting and Reconnecting the Account Synchronization DM to the Queue

You can prevent the Account Synchronization DM from enqueuing business events when you tune or shut down the queuing database schema by using the **pin_ctl** utility.

To disconnect from the queue, enter the following command:

```
pin_ctl stop dm_ifw_sync
```

To reconnect to the database queue and begin enqueuing business events, enter the following command:

```
pin_ctl start dm_ifw_sync
```

Disconnecting and Reconnecting DAT_Listener to the Queue

You can keep Pipeline Manager online when you tune or shut down the queuing database schema by disconnecting the DAT_Listener module from the queue.

To disconnect the module from the database queue, enter the following semaphore file entry:

```
ifw.DataPool.Listener.Module.Disconnect{}
```

To reconnect the DAT_Listener module to the database queue, enter the following semaphore file entry:

```
ifw.DataPool.Listener.Module.Connect{}
```

Troubleshooting Account Synchronization

If an error occurs during an account synchronization operation, check the account synchronization log file (*BRM_home*/**sys/dm_ifw_sync/dm_ifw_sync.pinlog**) for error codes.

Database Queue Creation Error

To install the Account Synchronization DM and create database queues, the **pin** user must have Oracle AQ privileges. If **pin** does not have privileges, you receive the following error when you attempt to install the Account Synchronization DM or create queues with the **pin_ifw_sync_oracle** utility:

```
PLS-00201 identifier 'SYS.DBMS_AQ' must be declared
```

To fix this error:

1. Using SQL*Plus, log in to your database as the SYS user and grant advanced queuing privileges to user **pin**:

   ```
   % sqlplus sys@databaseAlias
   Enter password: password

   SQL> grant execute on dbms_aq to pin;

   Grant succeeded.

   SQL> grant execute on dbms_aqadm to pin;

   Grant succeeded.

   SQL> grant execute on dbms_lock to pin;

   Grant succeeded.
   ```

2. Reinstall the Account Synchronization DM or create your queue by running the **pin_ifw_sync_oracle** utility manually.

See "Creating Additional Account Synchronization Queues".

Interleaved Processing Errors

This section describes interleaved processing errors and their solutions.

Missing Registry Entries

If you enable interleaved event processing in DAT_Listener, and you specify the event and CDR processing time entries (**MaxEventProcessTime** and **MaxCDRProcessTime**), you must also specify the entries that set the number of event thresholds (**MaxNumEvents** and **MinNumEvents**). If you do not do this, DAT_Listener throws a critical error (ERR_REG_VALUE_INVALID) during startup. However, if you update the registry by using a semaphore and the required entries are not specified, DAT_Listener throws a warning and disregards the semaphore update.

Semaphore Entry Errors

When an interleaving semaphore entry contains an error, the pipelines are deactivated.

The following interleaving semaphore errors cause the pipelines to be deactivated:

- The values for **MinNumEvents** and **MaxNumEvents** are not numbers.
- **MinNumEvents** is less than **MaxNumEvents**.
- The values for **MaxEventProcessTime** and **MaxCDRProcessTime** are not numbers.
- **CheckInterval** is not a number.

To reactivate the pipelines, use the **Active** semaphore. For example:

```
ifw.Pipelines.PRE_PROCESS.Active=TRUE
ifw.Pipelines.PRE_RECYCLE.Active=TRUE
ifw.Pipelines.ALL_RATE.Active=TRUE
```

After the pipelines are reactivated, reissue the interleaving semaphore using the correct values.

Modifying Business Events before Sending Them to Pipeline Manager

You can modify the BRM events that make up a business event before the business event is sent to the Account Synchronization DM for publishing to Pipeline Manager. You do this by customizing the Account Synchronization policy opcode, PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT. See *BRM Developer's Reference*.

> **Note:**
>
> Customizing the policy opcode requires programming knowledge and should be performed only by a developer.

You might want to modify an event to filter out unneeded data, which can improve performance if you publish large quantities of events. You can also use a flag to specify how adjustments

should be applied to the account balance; for example, to permit the account to have a negative balance.

To modify a business event before sending it to Pipeline Manager:

1. Ensure that all the BRM events that make up the business event are associated with opcode number 3626 (PCM_OP_IFW_SYNC_PUBLISH_EVENT) in your system's event notification list.

   See "Configuring Event Notification for Account Synchronization".

2. Use PCM_OP_IFW_SYNC_PUBLISH_EVENT to process the BRM events that make up the business event.

   See "Processing BRM Events That Make Up Account Synchronization Business Events".

3. Use the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode to customize the BRM events that make up the business event.

   See "Modifying BRM Events That Make Up Account Synchronization Business Events".

Processing BRM Events That Make Up Account Synchronization Business Events

PCM_OP_IFW_SYNC_PUBLISH_EVENT passes BRM events that make up Account Synchronization business events to the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode for modification.

PCM_OP_IFW_SYNC_PUBLISH_EVENT is called by the event notification feature when an event associated with this opcode in your system's event notification list occurs. See "Configuring Event Notification for Account Synchronization".

By default, PCM_OP_IFW_SYNC_PUBLISH_EVENT does not modify events; it only passes them to the policy opcode.

If the BRM event is published in the business event, PCM_OP_IFW_SYNC_PUBLISH_EVENT returns one of two POIDs:

- If the object passed in was an **/event** type object, the event POID is returned.
- If the object passed in was *not* an **/event** type object, a POID of type **/publish** is returned.

If the BRM event is not published (for example, if you filter out the event based on your business logic), the input flist is returned.

Modifying BRM Events That Make Up Account Synchronization Business Events

You can customize the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode to remove or enhance BRM events and event fields before they are assembled into business events and sent to Pipeline Manager. For example, you can customize this opcode to:

- Filter out BRM events that you do not want to include in published business events for various business or performance reasons.

> **Note:**
>
> – Do not filter out balance impact fields based on the balance element ID. If you do, you might remove fields needed by Pipeline Manager.
>
> – Do not modify the code that prepares the login fields (the **fm_ifw_sync_pol_prep_logins** function). This code is required when a login is changed.

- Modify the value of the PIN_FLD_FLAGS field that is added to PIN_FLD_BAL_IMPACT arrays. This field specifies how to apply adjustments. For example, you can permit a negative balance by specifying a value of **4**. By permitting a negative balance, the negative amount will be deducted from the account balance with the next billing cycle.

  You might use this method, for example, when a customer purchases a service that includes 60 free minutes per month and cancels the service before the end of the month. If the customer has used all 60 minutes, but you prorate the free minutes, the amount of usage for the canceled period can be deducted with the next billing cycle (provided the customer has a positive balance at that time).

By default, the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode modifies the following BRM events listed in Table 77-4:

Table 77-4 BRM Events Modified by PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT

Event	Action
/event/billing/debit	Adds a PIN_FLD_FLAGS field to each PIN_FLD_BAL_IMPACT array. This field indicates how adjustments should be applied: • **1** applies the entire adjustment to the current month. This flag is the default for cycle forward events. • **2** applies an adjustment over a period of months. **Note:** This flag value is not implemented. • **4** permits a negative account balance. Use this flag when you want to specify a negative balance for balance elements, such as when prorating free minutes. If this field is not set, any balance that results in a negative value is removed. • To specify more than one option, sum the values. For example, to apply the entire adjustment to the current month (**1**) and allow a negative balance (**4**), set this flag to **5**.
/event/billing/product/fee/cycle/*	Filters out these event types after the first occurrence. These events are generated when a charge offer is purchased and when billing is run. For charge offer purchase, Pipeline Manager must be notified so it can update the in-memory account data. However, for billing, Pipeline Manager performs the account balance update and does not need this event passed in.
/event/billing/sub_bal_validity	Creates a PIN_FLD_SUB_BAL_IMPACTS array in the event. PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT populates this array with information from element **1** of the existing PIN_FLD_SUB_BAL_AUDIT array for the event. Element **1** stores the updated validity period and the balance element and balance group associated with the sub-balance. Adding PIN_FLD_SUB_BAL_IMPACTS with the new validity period enables the DAT_BalanceBatch module to dynamically update the rating basis.
/event/customer/login	Copies PIN_FLD_SERVICE_OBJ from the top level of the input flist into the new login field (element **1**) in the PIN_FLD_LOGINS array. This ensures that Pipeline Manager knows which service is associated with the new login.

If the BRM event is published in the business event, the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode returns one of the following POIDs:

- If the object passed in was an **/event** type, the event POID is returned.
- If the object passed in was *not* an **/event** type object, a POID of type **/publish** is returned.

If the BRM event is not published (for example, if you filter out the event based on your business logic), the input flist is returned.

Manually Configuring Object Auditing

To enable BRM to audit GSM objects, run the **object_auditing.pl** script. For more information, see "Turning On Object Auditing".

The **object_auditing.pl** script performs a series of steps that can also be performed manually. This section describes these manual steps.

To manually configure object auditing:

- Modify the DM configuration file (**pin.conf**) and run the **pin_history_on** utility.

  See "Running the pin_history_on Utility".
- Modify the **create_indexes_audit_tables_oracle.source** file and create the audit table indexes.

  See "Creating Audit Table Indexes".

Running the pin_history_on Utility

> **Caution:**
>
> - When installing wireless managers such as Account Synchronization Manager or GSM Manager, you must install *all* managers *before* running this utility. If you run this utility before installing a wireless manager, your installation will fail.
> - If you are using a multischema system, do the following:
>
>   1. Install all wireless managers.
>   2. Install your multischema system.
>   3. Run the **pin_history_on** utility.

> **Note:**
>
> - The **pin_history_on** utility requires a **pin.conf** configuration file. You can create one, or you can run the script from a directory that contains a **pin.conf** file, such as *BRM_home*/**sys/dm_ifw_sync**.
> - Before running this utility, you must manually modify entries in the Oracle DM configuration file to give the DM write permission so that the objects in the input file can be written to the database. After running the utility, you restore the entries in the configuration file to their original values.

To run the **pin_history_on** utility:

1. Open the *BRM_home*/**sys/dm_oracle/pin.conf** file in a text editor.
2. Note the value of the following entries:

```
- dm dd_write_enable_fields
- dm dd_write_enable_objects
- dm dd_write_enable_portal_objects
- dm dd_mark_as_portal
```

3. Set the values of the following entries to **1**.

```
- dm dd_write_enable_fields 1
- dm dd_write_enable_objects 1
- dm dd_write_enable_portal_objects 1
- dm dd_mark_as_portal 1
```

> **Note:**
>
> If an entry is not in the file, add it.

4. Save and close the file.
5. Stop and restart the Oracle DM.
6. Run the **pin_history_on** utility:

```
pin_history_on pin_history_on_input
```

> **Note:**
>
> If you run the utility from another directory, include the path to the **pin_history_on** utility in the command line. For example:
>
> **pin_history_on** *BRM_home*/**bin/pin_history_on_input**

7. Open the *BRM_home*/**sys/dm_oracle/pin.conf** file in a text editor.
8. Restore the following entries to their original values (the values they had before you modified them):

```
- dm dd_write_enable_fields
- dm dd_write_enable_objects
```

```
- dm dd_write_enable_portal_objects
- dm dd_mark_as_portal
```

9. Save and close the file.
10. Stop and restart the Oracle DM.

Creating Audit Table Indexes

After running the **pin_history_on** utility, you must create indexes for the new audit tables.

The Account Synchronization installation installs a file (**create_indexes_audit_tables_oracle.source**) that specifies the necessary indexes on the audit tables.

1. Open the *BRM_home***/sys/dd/data/create_indexes_audit_tables_oracle.source** file in a text editor.
2. Set the $PIN_CONF_TBLSPACEX1 entry to **pinx00**.

> **Note:**
>
> This entry must have the same value as the variable **$MAIN_DB{'indexes_group'}** configured in the *BRM_home***/setup/pin_setup.values** file. The default value of this entry is **pinx00**.

3. Set the $PIN_CONF_STORAGE_MED entry to the following:

```
storage (initial 200k next 200k maxextents unlimited pctincrease 0)
```

4. Save and close the file.
5. Create the audit table indexes in SQL*Plus by running the following command:

```
% sqlplus user@database_Alias
Enter password: password

SQL> @filepath/create_indexes_audit_tables_oracle.source
```

Synchronizing Balance Group Transfer Data with Pipeline Manager

If your system uses both real-time rating and batch rating, you must configure BRM to notify Pipeline Manager when:

- A service transfers to a different balance group.
- A balance group transfers to a different bill unit.

BRM notifies Pipeline Manager through the **ServiceBalanceGroupTransfer** business event.

To configure BRM to notify Pipeline Manager when a service transfers to a different balance group, install the Account Synchronization DM and configure it to publish the **ServiceBalanceGroupTransfer** business event to the account synchronization queue.

When configuring the Account Synchronization DM:

- Make sure the **ServiceBalanceGroupTransfer** business event is listed in your payload configuration file under the **<PublisherDefs>** section. This business event appears in the

default Account Synchronization payload configuration file (*BRM_home*/**sys/eai_js/payloadconfig_ifw_sync.xml**).

- Make sure the event notification list maps the **/event/notification/service_balgrp_transfer/data notification** event to opcode number **3626**. This mapping appears in the default Account Synchronization event notification file (*BRM_home*/**sys/data/config/pin_notify_ifw_sync**).
- Add the **ServiceBalanceGroupTransfer** business event to your *BRM_home*/**sys/dm_aq/aq_queuenames** file, if the file is configured to send specific business events.

78

Account Synchronization Installation Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Account Synchronization installation utilities.

object_auditing

Use this utility to enable object auditing when using the Oracle database.

> **Caution:**
>
> - When installing wireless managers such as Account Synchronization Manager or GSM Manager, you must install *all* managers *before* running this utility. If you run this utility before installing a wireless manager, your installation will fail.
> - If you are using a BRM multischema system, you must install all wireless managers, then install your multischema system, and then run the **object_auditing.pl** utility, in that order.

The **object_auditing.pl** script turns on object auditing by invoking the **pin_history_on** utility with the **-v** and **pin_history_on_input** input file parameters to turn on object auditing. The script then creates the specified audit table indexes by using the **create_indexes_audit_tables_oracle.source** file.

You define which objects are audited by editing the **pin_history_on_input** file (*BRM_home***/apps/integrate_sync/pin_history_on_input**) before running this utility to load the file.

You customize the audit table indexes by editing the **create_indexes_audit_tables_oracle.source** file (*BRM_home***/sys/dd/data/create_indexes_audit_tables_oracle.source**).

> **Note:**
>
> To connect to the BRM database, the **object_auditing** script needs a configuration file in the directory from which you run the utility.

Location

*BRM_home***/setup/scripts**

Syntax

```
perl object_auditing
```

Results

Look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_history_on

Use this utility to enable object auditing when using the Oracle database.

Use the **object_auditing.pl** script to run this utility after installing all your wireless managers. See "object_auditing".

> **Caution:**
>
> - When installing wireless managers such as Account Synchronization Manager or GSM Manager, you must install *all* managers *before* running this utility. If you run this utility before installing any wireless manager, your installation will fail.
> - If you are using a multischema system, you must install all wireless managers, then install your multischema system, and then run the **pin_history_on** utility, in that order only.

If you choose to run this utility independently (not as part of the **object_auditing** script), you must also perform these tasks:

- Manually modify entries in the Oracle DM configuration file to give the DM permission to write the objects in the input file to the database.
- Create audit table indexes for the objects that you audit.

You define which objects are audited by editing the **pin_history_on_input** file and running this utility to load the file.

> **Note:**
>
> To connect to the BRM database, the **pin_history_on** utility needs a configuration file in the directory from which you run the utility. S

Location

*BRM_home***/bin**

Syntax

```
pin_history_on [-d | -v | -h] pin_history_on_input
```

Parameters

-d
Enables debugging mode.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **pin_history_on** *any_other_parameter* **-v>** *filename.***log**

-h
Displays the syntax and parameters for this utility.

pin_history_on_input
Name of the file that specifies which fields within the objects to audit. A sample file is included in the *BRM_home***/apps/integrate_sync** directory

Results

Look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_ifw_sync_oracle

Use this utility to create, drop, and monitor Account Synchronization queues in your Pipeline Manager database.

The Account Synchronization DM uses these queues to send BRM business events to the Pipeline Manager Listener (DAT_Listener) module.

Location

*BRM_home***/apps/pin_ifw_sync**

Syntax Overview

The following actions are supported for Pipeline Manager databases:

- Syntax for Creating a Queue
- Syntax for Dropping a Queue
- Syntax for Generating a Report
- Syntax for Testing a Queue
- Syntax for Listing Queues
- Syntax for Getting Help

Syntax for Creating a Queue

Creates an Account Synchronization queue, queue table, and database package in your database. The database package contains stored procedures for queuing, dequeuing, and purging business events.

```
pin_ifw_sync_oracle.pl create [-l /@DatabaseAlias]
                              [-q queue_name -t queue_table]
                              [-s]storage_clause
                              [-r retention_time]
```

Parameters for Creating a Queue

-l /@*DatabaseAlias*
Specifies how to connect to the database.
For example:

```
pin_ifw_sync_oracle.pl create -l /@pindb.example.com
```

If you omit this parameter, the utility prompts you for this information.

-q *queue_name* -t *queue_table*
Specifies the queue name and queue table name.
If you omit these parameters, the utility automatically creates a queue named IFW_SYNC_QUEUE and a queue table named IFW_SYNC.

-s *storage_clause*
Specifies the storage settings for the queue table.
If you omit this parameter, the storage settings are set by the **storage_clause** parameter in the *BRM_home*/**apps/pin_ifw_sync/create_ifw_sync_queue.conf** file.
For example:

```
pin_ifw_sync_oracle.pl create -s "tablespace PIN00 initrans 5 storage (initial 200k
next 200k maxextents unlimited pctincrease 0 )"
```

-r *retention_time*
Specifies the amount of time, in seconds, until processed events are removed from the database queue.
If you omit this parameter, the retention time is set by the **retention_time** parameter in the *BRM_home*/**apps/pin_ifw_sync/create_ifw_sync_queue.conf** file.

Syntax for Dropping a Queue

Drops the specified queue and its associated queue table from your database. If the database contains no other Account Synchronization queues, this command also drops the Account Synchronization database package, which contains stored procedures for queuing, dequeuing, and purging events.

```
pin_ifw_sync_oracle.pl drop [-q queue_name] [-l /@DatabaseAlias]
```

Parameters for Dropping a Queue

-q *queue_name*
Specifies the name of the queue to drop.
If you omit this option, the utility automatically drops the default queue, IFW_SYNC_QUEUE.

-l /@*DatabaseAlias*
Specifies how to connect to the database.
If you omit this parameter, the utility prompts you for this information.

Syntax for Generating a Report

Generates a report that displays the state of each event in an Account Synchronization queue.

```
pin_ifw_sync_oracle.pl report  -r summary|detail [-q queue_name]
                               [-l /@DatabaseAlias]
```

Parameters for Generating a Report

-r summary | detail
Generates the specified type of report.

- **-r summary** generates a report that summarizes the number of events in each state. Events can be in the following states shown in Table 78-1:

State	Description
READY	The event has not been dequeued or processed by Pipeline Manager.
PROCESSED	The event was dequeued and processed by Pipeline Manager.

- **-r detail** generates a report that details the ID, event state, queuing time, and dequeuing time for each event.

-q *queue_name*
Specifies the queue name.
If you omit this parameter, the utility automatically generates a report for the default queue, IFW_SYNC_QUEUE.

-l /@*DatabaseAlias*
Specifies how to connect to the database.
If you omit this parameter, the utility prompts you for this information.

Syntax for Testing a Queue

Tests the specified queue by attempting to enqueue and dequeue 20 test events. You run this command to test if a newly created queue functions properly.

> **Note:**
>
> You need to test a queue only after it is first created.

```
pin_ifw_sync_oracle.pl test [-q queue_name] [-l /@DatabaseAlias]
```

Parameters for Testing a Queue

-q *queue_name*
Specifies the queue name.
If you omit this parameter, the utility automatically tests the default queue, IFW_SYNC_QUEUE, and default queue table, IFW_SYNC.

-l /@*DatabaseAlias*
Specifies how to connect to the database.
If you omit this parameter, the utility prompts you for this information.

Syntax for Listing Queues

Lists all queues in the current user's database.

```
pin_ifw_sync_oracle.pl list [-l /@DatabaseAlias]
```

Parameters for Listing Queues

-l /@*DatabaseAlias*
Specifies how to connect to the database.
If you omit this parameter, the utility prompts you for this information.

Syntax for Getting Help

Displays the syntax for the **pin_ifw_sync_oracle** utility.

```
pin_ifw_sync_oracle.pl help
```

Results

The **pin_ifw_sync_oracle** utility notifies you when it runs successfully. Otherwise, look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

Part XI

Pipeline Manager Reference

This part provides reference information about Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager. It contains the following chapters:

79

BRM Rating EDR Container Description

This chapter describes the rating EDR container fields that are used by Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Naming Conventions

The following definitions listed in Table 79-1 are used in this document to describe the record format:

Table 79-1 Record Format Definitions

Symbol	Meaning
X	Alphanumeric, left-justified, filled with trailing spaces to the right.
Z	Numeric, left-justified, filled with trailing spaces to the right.
H	Hexadecimal value (0-9, A-F), right-justified, filled with leading zeros to the left.
9	Numeric, right-justified, filled with leading zeros to the left.
(m)	Specifies the length in characters: mandatory.
[n]	Specifies the decimal precision. Optional.

Oracle CDR Format

The Oracle CDR format is the standard file structure used by Pipeline Manager to process CDRs during the input and output processes.

The Oracle CDR format has the following characteristics:

- Each record is separated by a newline character (**\n**).
- Each record contains data for one service only.
- Each record contains a fixed number of fields.
- Each field is tab delimited (**\t**).
- Each field is in a specified position within a record. For example, the first field in the Header Record is the Record Type, the second is the Sender, and so forth.

> **Note:**
>
> If a field does not have a value, the field is left blank. The result is a tab followed by another tab.

- Each field has a specified data type and format. For example, the **A number** must be a string that is 10 characters long**.**

To process CDRs, Pipeline Manager converts the CDRs to the internal EDR format by using the stream format, grammar, and mapping description files.

The stream format file describes the structure of the Oracle CDR format.

The following example shows the section of the stream format description file that describes the format of the Header Record. The Header Record is identified by the record type **010**. The fields are separated by a tab (**\t**), and the record is terminated by a newline character (**\n**). It specifies the list the fields in the record. The first field is the record type (RECORD_TYPE), the second is the record number (RECORD_NUMBER), and so forth. RECORD_TYPE uses the AscString() data type, RECORD_NUMBER uses the AscInteger( ) data type.

```
HEADER(SEPARATED)
{
Info
{
Pattern = "010.*\n";
FieldSeparator = '\t';
RecordSeparator = '\n';
}
RECORD_TYPE                     AscString();
RECORD_NUMBER                   AscInteger();
SENDER                          AscString();
RECIPIENT                       AscString();
SEQUENCE_NUMBER                 AscInteger();
ORIGIN_SEQUENCE_NUMBER          AscInteger();
CREATION_TIMESTAMP              AscDate();
TRANSMISSION_DATE               AscDate("%Y%m%d");
TRANSFER_CUTOFF_TIMESTAMP       AscDate();
UTC_TIME_OFFSET                 AscString();
SPECIFICATION_VERSION_NUMBER    AscInteger();
RELEASE_VERSION                  AscInteger();
ORIGIN_COUNTRY_CODE              AscString();
SENDER_COUNTRY_CODE              AscString();
DATA_TYPE_INDICATOR              AscString();
IAC_LIST                         AscString();
CC_LIST                          AscString();
UTC_END_TIME_OFFSET              AscString();
}
```

The grammar files are used to verify the data formats and to normalize the data. For example, if a field is supposed to be 10 characters, Pipeline Manager uses the grammar file to perform this check. If the data is of an incorrect format, the CDR is rejected.

The mapping files are used to map CDR fields to the EDR container fields.

The following example shows a section of the InMap description file, which is used during the input process. This example shows how the fields in the Header Record of a CDR are mapped to the EDR container fields.

```
HEADER
{
STD_MAPPING
{
RECORD_TYPE                 -> HEADER.RECORD_TYPE;
RECORD_NUMBER               -> HEADER.RECORD_NUMBER;
SENDER                      -> HEADER.SENDER;
RECIPIENT                   -> HEADER.RECIPIENT;
SEQUENCE_NUMBER             -> HEADER.SEQUENCE_NUMBER;
ORIGIN_SEQUENCE_NUMBER      -> HEADER.ORIGIN_SEQUENCE_NUMBER;
CREATION_TIMESTAMP          -> HEADER.CREATION_TIMESTAMP;
TRANSMISSION_DATE           -> HEADER.TRANSMISSION_DATE;
```

```
TRANSFER_CUTOFF_TIMESTAMP    -> HEADER.TRANSFER_CUTOFF_TIMESTAMP;
UTC_TIME_OFFSET              -> HEADER.UTC_TIME_OFFSET;
SPECIFICATION_VERSION_NUMBER -> HEADER.SPECIFICATION_VERSION_NUMBER;
RELEASE_VERSION              -> HEADER.RELEASE_VERSION;
ORIGIN_COUNTRY_CODE          -> HEADER.ORIGIN_COUNTRY_CODE;
SENDER_COUNTRY_CODE          -> HEADER.SENDER_COUNTRY_CODE;
DATA_TYPE_INDICATOR          -> HEADER.DATA_TYPE_INDICATOR;
IAC_LIST                     -> HEADER.IAC_LIST;
CC_LIST                      -> HEADER.CC_LIST;
UTC_END_TIME_OFFSET          -> HEADER.UTC_END_TIME_OFFSET;
}
}
```

EDR Format Structure

The BRM EDR format consists of the following components:

1. Exactly one Header Record. Record type 010.
2. Zero or more Basic Records in no specific order:
   a. Basic Detail Record; for example, record type 020.
   b. More basic records might be defined in the future.
3. Zero or more Associated Records, related to one Basic Record, in the following order:
   a. Associated Service Extension Records; for example, record type 520.
   b. Associated CAMEL Extension Records. Record type 700.
   c. Associated BRM Balance Record. Record type 900.
   d. Associated Zone Breakdown Record; for example, record type 960.
   e. Associated Charge Breakdown Record; for example, record type 981.
   f. Associated Message Description Record. Record type 999.
4. Exactly one Trailer Record. Record type 090.

Example Structure

Table 79-2 contains an example BRM EDR structure.

Table 79-2 BRM EDR Example Structure

Record	Description
Header Record: 010	Once. Mandatory.
Basic Detail Record: 040	Once. Optional.
Associated GPRS Extension Record: 540	Once. Optional.
Associated CAMEL Extension Record: 700	Once. Optional.
Associated BRM Balance Record: 900	Once. Optional.
Supplementary Balance Impact Packet Record: 600	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Impact Packet Record: 605	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Info Packet Record: 607	*n* times. Mandatory 1-*n*.
Associated Zone Breakdown Record: 961	*n* times. Optional.

Table 79-2 (Cont.) BRM EDR Example Structure

Record	Description
Supplementary Zone Packet Record: 660	*n* times. Mandatory 1-*n*.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Supplementary Charge Packet Record: 660	*n* times. Mandatory 1-*n*.
Associated Message Description Record: 999	*n* times. Optional.
Basic Detail Record: 070	Once. Optional.
Associated WAP Extension Record: 550	Once. Optional.
Associated BRM Balance Record: 900	Once. Optional.
Supplementary Balance Impact Packet Record: 600	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Impact Packet Record: 605	*n* times. Mandatory 1-*n*.
Supplementary Sub-Balance Info Packet Record: 607	*n* times. Mandatory 1-*n*.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Supplementary Charge Packet Record: 660	*n* times. Mandatory 1-*n*.
Basic Detail Record: 021	Once. Optional.
Associated GSM Extension Record: 520	Once. Optional.
Supplementary Service Event Record: 520	*n* times. Optional.
Basic Service Event Record: 520	*n* times. Optional.
Associated CAMEL Extension Record: 700	Once. Optional.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Supplementary Charge Packet Record: 660	*n* times. Mandatory 1-*n*.
Basic Detail Record: 127	Once. Optional.
Associated Charge Breakdown Record: 981	*n* times. Optional.
Basic ...	None
Associated …	None
Supplementary …	None
Trailer Record: 090	Once. Mandatory.

Expected File Name

The BRM EDR file name uses a format of SOL42_*SenderRecipientSequence_number*.DAT. Table 79-3 describes the attributes used in the file name.

Table 79-3 EDR File Name Attributes

Item	Format	Description
Sender	X(5)	code for the sender of the file (for example, D00D1)
Recipient	X(5)	code for the recipient of the file (for example, SOL42)
Sequence_number	9(6)	sequence number of the file (000000 to 999999)

Example: "SOL42_D00D1SOL42004711.DAT"

Record Type Ranges

The Record Type Ranges listed in Table 79-4 are defined:

Table 79-4 Record Type Ranges

Range	Record type
000 - 009	Reserved for internal usage
010	Header Record
011 - 019	Reserved for Basic Address Records
020 - 089	Basic Detail Records
090	Trailer Record
091 - 099	Reserved for internal usage
100 - 299	Basic Detail Records
300 - 319	Basic Recharge Records
320 - 399	Free
400 - 499	Reserved for further Basic Record Types
500 - 599	Associated Service Extension Records
600 - 699	Supplementary Records (for former Sub-Blocks of Associated Records)
700 - 749	Associated CAMEL / IN Records
750 - 799	Reserved for further Associated Record Types
800 - 899	Free
900 - 949	Associated Balance Records
950 - 959	Reserved for further Record Types
960 - 969	Associated Zone Breakdown Records
970 - 998	Associated Charge Breakdown Records
999	Associated Message Description Record

> **Note:**
>
> Not all of the given Record Types have been defined. Undefined values are reserved for future use.

Header Record (RECType 010)

This record is always the first record within a file. Table 79-5 describes the fields in the Header Record.

Table 79-5 Header Record Fields

Name	Format	Description
RECORD_ LENGTH	Integer	Optional for backward compatibility.

Table 79-5 (Cont.) Header Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long, first byte denotes the market; for example, GSM, ISDN.010. Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of the record in the file. Ensures a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. Always **000000001**.
SENDER	X(10)	Unique identifier of the PLMN or physical (network) operator, which is sending the file; used to determine the network, which is the sender of the data. The full list of mobile codes in use is given in MoU TADIG PRD TD. 13: PLMN Naming Conventions. Specifies a unique NOSP_ID together with the RECIPIENT. Can also be used to determine the network operator responsible for the CDRs. Derivation: Optional, but should be defaulted if not present on the input side, for example, by own NO-Id, for example, 'DTAG'. Set by the first processor and left unchanged.
RECIPIENT	X(10)	Unique identifier of the PLMN or physical (network) operator to whom the file is being sent. See the MoU TADIG PRD TD. 13: PLMN Naming Conventions for a list of mobile codes. Specifies a unique NOSP_ID together with the SENDER. Can also be used to determine the reseller or service provider who is responsible for billing these events. Derivation: Optional, but should be defaulted; for example, by your own NO-Id, such as 'DTAG'. Set by the first processor and left unchanged.
SEQUENCE_NUMBER	9(6)	Unique reference that identifies each file sent by the VPLMN or logical sender to a particular HPLMN or logical recipient. It indicates the file number of the specific file type, starting at 1 and increments by one for each new file of that type sent. Separate sequence numbering must be used for test and chargeable data. Having reached the maximum value (999999), the number restarts at 1. Validates duplicate sequence numbers and sequence number gaps. **Note:** In the case of retransmission, this number does not increment. Range: 000001 - 999999 for test data and chargeable data. Derivation: Optional, if no sequence check is performed. Mandatory, if a sequence check is performed. Should be set by the first processor and can be changed by any following processor; for example, in case of recycling to assure a unique and linear sequence order to all following processors.

Table 79-5 (Cont.) Header Record Fields

Name	Format	Description
ORIGIN_SEQUENCE_NUMBER	9(6)	Original file sequence number as generated the first time. Identical content as SEQUENCE_NUMBER, but will never be changed. Used as a reference to the original file, if any processor has changed the file sequence number. Derivation: Mandatory, defaulted by SEQUENCE_NUMBER. Set by the first processor and left unchanged.
SEQ_CHECK_KEY	String	Derivation: Optional if no sequence check is performed. Mandatory if a sequence check is performed.
SEQ_GEN_KEY	String	Derivation: Optional if no sequence check is performed. Mandatory if a sequence check is performed.
CREATION_TIMESTAMP	YYYYMMDDHHMISS	Date and time on which the file was created. Not required by GSM MoU BA. 12, but might be useful for operational purposes. Can be used to validate that at least one file/stream has been generated every day. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory, defaulted with the FILESYSTEM-SYSDATE or a Transaction-Start-Timestamp. Set by the first processor and left unchanged.
TRANSMISSION_DATE	YYYYMMDD	Date on which the file was sent from the sender network to the recipient network or data clearing house. Can be used to calculate the run time of a file/stream between creation and transmission. Also used as a default TRANSFER_CUTOFF_TIMESTAMP. Derivation: Mandatory, defaulted with SYSDATE. Set by the first processor and left unchanged.
TRANSFER_CUTOFF_TIMESTAMP	YYYYMMDDHHMISS	Date and time used to select calls for transfer. All records available prior to the timestamp are transferred. This gives an indication to the recipient as to how current the information is. Can be used to validate that all CDRs are prior to this date and time. Their CHARGING_START_TIMESTAMP must be equal or less. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory, defaulted with TRANSMISSION_DATE. Set by the first processor and left unchanged.

Table 79-5 (Cont.) Header Record Fields

Name	Format	Description
UTC_TIME_OFFSET	X(5)+/-HHMI	All timestamps are sender (VPLMN) local time. So that the time can be equated to time in the recipient (HPLMN) local time, the sender gives the difference between local time and UTC time. UTC Time Offset = Local Time minus UTC Time. Can be used to translate the TRANSFER_CUTOFF_TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Example: Washington DC, USA 1000hrs10/10/97 UTC Time1500hrs10/10/97 UTC Time Offset= 10 - 15 = -0500 Madrid, Spain1600hrs10/10/97 UTC Time1500hrs10/10/97 UTC Time Offset= 16 - 15 = +0100 **Note:** Where dates are different, 24 is added to the time of the greater date. Derivation: Mandatory. Set by the first processor and left unchanged.
SPECIFICATION_VERSION_NUMBER	9(2)	Uniquely identifies the format. Different specification versions indicate that the record structure has changed; for example, field length, new fields, and new record types. Used for encoding different formats. Range: 01 Derivation: Mandatory. Set by the first processor and left unchanged.
RELEASE_VERSION	9(2)	Indicates the release version within the Specification Version Number. Different Release Versions indicates that only the content of fields has changed. Used for encoding different formats. Derivation: Mandatory. Set by the first processor and left unchanged.
ORIGIN_COUNTRY_CODE	X(8)	International access and country code, which applies within the country of the network where the CDR originated. Might be useful for an international billing center to distinguish between national and international calls; for example, within the basic detail record. Range: 0049; for example, for Germany. Derivation: Mandatory. Set by the first processor and left unchanged.

Table 79-5 (Cont.) Header Record Fields

Name	Format	Description
SENDER_COUNTRY_CODE	X(8)	International access and country code that applies within the country of the sender (VPLMN). This might be different from the originating code if the sender is a clearing house or third-party operator. Might be useful for an international billing center. Range: 0049; for example, for Germany Derivation: Mandatory. Set by the first processor and left unchanged.
DATA_TYPE_INDICATOR	X(1)	The type of data contained within the file; for example, test or chargeable data. Any customer billing processor should ignore test data or at least separate these streams. Values: T: Test Data Space: Chargeable Data Derivation: Mandatory. Set by the first processor and left unchanged.
IAC_LIST	X(30)	Comma-separated list of all international access codes used within this file. Used during number normalization to detect numbers already starting with these IACs. Those numbers will not be normalized anymore. Example: "001,002" for two IACs Derivation: Optional. Set by the first processor and left unchanged.
CC_LIST	X(30)	Comma-separated list of all country codes used within this file. Used during number normalization to detect all numbers already starting with these CCs. Those numbers are normalized by adding a default IAC. Example: "49,33,1" for two CCs Derivation: Optional. Set by the first processor and left unchanged.
TAP_DECIMAL_PLACES	Integer	Derivation: Optional, but mandatory for RAP output. Set by the input grammar.
OPERATOR_SPECIFIC INFO	String	Derivation: Optional, default = '' Stores a key that identifies the CDR used to generate a specific EDR. Useful for RAP or CIBER return. Must be set by an iScript.
CIBER_FILLER	String	Optional.
CIBER_RECORD_TYPE	String	Optional. See CIBER specs for usage.

Table 79-5 (Cont.) Header Record Fields

Name	Format	Description
RETURN_INDICATOR	String	Optional. See CIBER specs for usage.
CURRENCY	String	Optional. See CIBER specs for usage.
SETTLEMENT_PERIOD	String	Optional. See CIBER specs for usage.
CLEARINGHOUSE_ID	String	Optional. See CIBER specs for usage.
BATCH_REJECT_REASON	String	Optional. See CIBER specs for usage.
BATCH_CONTENTS	String	Optional. See CIBER specs for usage.
SENDING_CLEARINGHOUSE_BID	String	Optional. See CIBER specs for usage.
CREATION_PROCESS	String	Process that created output stream.
SCHEMA_VERSION	String	Version number for schema.
EVENT_TYPE	String	BRM event type.
RAP_FILE_SEQ_NO	String	Optional. Indicates the returned account procedure (RAP) file in which the recipient public data network (PMN) returned the TAP file batch to the sender PMN. This field is a unique reference. Used in TAP files.
QUERYABLE_FIELDS_MAPPING	String	Optional Calculated for suspense handling. Contains the database column names and data types that map to queryable fields. Use this format: *column_name*:*data_type*[;*column_name*:*data_type*[;...]]
BATCH_ID	String	Optional. Set to the actual file batch ID.
UTC_END_TIME_OFFSET	X(5)	Timezone where the call terminated. Derivation: Optional.
BATCH_CTRL_INFO_START_INDEX	Integer	BatchControlInfo block start index.
BATCH_CTRL_INFO_END_INDEX	Integer	BatchControlInfo block end index.
ACCOUNTING_INFO_START_INDEX	Integer	AccountingInfo block start index.
ACCOUNTING_INFO_END_INDEX	Integer	AccountingInfo block end index.
NETWORK_INFO_START_INDEX	Integer	NetworkInfo block start index.
NETWORK_INFO_END_INDEX	Integer	NetworkInfo block end index.
MESSAGE_DESCRIPTION_START_INDEX	Integer	MessageDescriptionInfoList block start index.

Table 79-5 (Cont.) Header Record Fields

Name	Format	Description
MESSAGE_DESCRIPTION_END_INDEX	Integer	MessageDescriptionInfoList block end index.
NOTIFICATION_START_INDEX	Integer	Notification block start index.
DELAYED_ERROR_BLOCK	String	Stores the block name that has the fatal error.
OBJECT_CACHE_TYPE	Integer	Cache residency type: • **0**: Convergent • **1**:- Prepaid • **2**: Postpaid
TAP_FILE_TYPE	String	Type of TAP file, TAP3 or TAP311.

Basic Detail Record (RECType 020-089, 100-299)

This record references a billable event. This basic record is the primary record within the BRM format structure. Table 79-6 lists the fields in the Basic Detail Record.

Table 79-6 Basic Detail Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. First byte denotes the market. 020 MOC Switch Mobile Originating Call 021 TA_MOC TAP Mobile Originating Call (Roaming**) 022 CFW Mobile Switch Call Forwarding 023 RCF/RFD Mobile Roaming Call Forwarding 024 SMO Mobile Short Message Originating 025 SMT Mobile Short Message Terminating 026 VMO Mobile Voice Mail Originating 027 OAB Mobile Operator Assisted Call (Basic) 028 OAS Mobile Operator Service (Call Completion) 029 MSS Mobile Supplementary Service Event 030 MTC Switch Mobile Terminating Call 031 TA_MTC TAP Mobile Termination Call (Roaming**)

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
RECORD_TYPE (cont.)	String	040 SGSN_MOC Serving GPRS Support Node Originating 041 SGSN_MOT Serving GPRS Support Node Terminating 042 GGSN_MOC Gateway GPRS Support Node Originating 043 GGSN_MOT Gateway GPRS Support Node Terminating 044 GPRS_SMO GPRS - Short Message Originating 045 GPRS_SMT GPRS - Short Message Terminating 046 HSCSD_MOC Mobile HSCSD Originating Call 047 HSCSD_MOT Mobile HSCSD Terminating Call 048 TA_GPRSOC TAP GPRS Originating (Roaming**) 049 TA_GPRSTC TAP GPRS Termination (Roaming**) 050 SCU Basic Service Center Usage Record 060 VAS Basic Value Added/Event Record 070 WAP Basic WAP Record 120 POC ISDN/Public Switch Originating 121 DX_POC ISDN/Public Switch Orig.(data exchange) 122 PCF ISDN/Public Switch Call Forwarding 126 PVM ISDN/Public Switch Voice Mail Originating 127 POB ISDN/Public Operator Assisted Call (Basic) 128 POS ISDN/Public Operator Service (Call Com-pl) 130 PTC ISDN/Public Switch Termination Call 131 DX_PTC ISDN/Public Switch Term. (data exchange) 220 IOCBasic Internet Record Other record types might be defined when necessary. Derivation: Mandatory. Set by the first processor and left unchanged. **Note:** Only record types 021 and 031 are treated as roaming calls.
RECORD_NUMBER	9(9)	Sequence number of record in file. Ensures a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. Important: Following modules might change this record number; for example, if new record types are inserted.
DISCARDING	9(1)	Indicates if an EDR should be discarded or rejected. Values: 0: Proceed (default) 1: Reject 2-9: Discard The values from 2 to 9 represent different discarding reasons. Derivation: Mandatory. Might be set by any processor.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CHAIN_REFERENCE	X(10)	Identifies an EDR as part of a long event that has been split into multiple EDRs. Condition: Only present if more than one record is raised for a call (default=Spaces). Value: Any six-digit number. Derivation: Optional. Might only be set by the first processor.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SOURCE_NETWORK_TYPE	X(1)	The source network type; for example, GSM 900. This is needed for specific implementation models such as some satellite operators where the network originating the chargeable record might be lost. **Note:** This is a temporary solution pending further developments. Values: Mobile-Networks: A: S-41 AMPS A B: S-41 AMPS B C: S-41 Satellite D: S-95 CDMA E: S-136 TDMA F: PDC G: GSM 900 H: GSM 1800 I: GSM 1900 J: GSM 90011800 K: GSM 90011900 L: GSM Satellite M: UMTS N: Telematic O: GPRS - GGSN P: GPRS - SGSN Intercarrier-Networks: W: Inroute X: Outroute T: Transit Z: undefined Fixed-Networks: 0: General Fixed Network 1: Analog 2: ISDN 3: ADSL 4: Multiplex Other-Networks: 9: Internet Other values might apply according to the related original input format. Derivation: Optional. Might be set by the first processor and might be changed by an interconnect rating processor.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SOURCE_NETWORK	X(14)	Network code from which the call or message was routed. This could either be PLMN_ID or any logical operator code. Used for interconnect rating. Condition: In case of interconnect rating it is overwritten by the network operator code related to the inroute. See TRUNK_INPUT. Derivation: Optional (only mandatory for the interconnect processor).
DESTINATION_NETWORK_TYPE	X(1)	Indicates the destination network type; for example, GSM 900. This is needed for specific implementation models such as some satellite operators where the network terminating the chargeable record might be lost. **Note:** This is a temporary solution pending further developments.Values: See SOURCE_NETWORK_TYPE. Derivation: Optional. Might be set by the first processor and might be changed by an interconnect rating processor.
DESTINATION_NETWORK	X(14)	Network towards which the call or message is routed. Condition: Where a short message has not been delivered or where optimal routing is not used, the field is set to spaces. In case of interconnect rating, it is overwritten by the network operator code related to the outroute. See TRUNK_OUTPUT. Derivation: Optional (only mandatory for interconnect rating).
TYPE_OF_A_IDENTIFICATION	X(1)	Specifies if the number used to identify the subscriber within the network is an IMSI or an MSISDN. This type does not relate to the A Number representation. Values: A: Internet or Account Number (default for internet) C: Calling Card Number I: IMSI M: MSISDN (default for fixed networks) P: IP Number S: SIM-ICC (default for GSM) X: undefined Derivation: Optional. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
A_MODIFICATION_INDICATOR	H(2)	Specifies if the called or calling number has been modified by the VPLMN; for example, for privacy reasons. Can be used to evaluate how to handle the number internally; for example, print the last three digits in clear text (anonymize) or suppress the complete CDR during printing a detailed invoice. Condition: PLMNs are not forced to implement this parameter. If not implemented, the number must not be modified. Values: 00: Default setting (undefined) and normal 01: Social number 02: Anonymized number 04: Special number (for example, premium rate) 08: Modified number (for example, vanity routing or short number translation) Derivation: Optional. Set by the first processor and left unchanged.
A_TYPE_OF_NUMBER	Z(1)	Type of address associated with a particular destination or calling number. Condition: Not all networks support this parameter. Values: 0: Nature of address unknown (default) 1: International number 2: National significant number 3: Network-specific number 4: Subscriber number 5: Abbreviated number Derivation: Optional, default=0. From bits 7 - 5 of octet 1 of the GSM Address String type as defined in TS GSM 09.02. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
A_NUMBERING_PLAN	X(1)	The numbering plan associated with a particular destination or calling number. Might be useful when analyzing the A Number for normalization reasons; for example, to interpret the structure of the number (for example, distinguish IP numbers. Condition: Not all networks support this parameter. Values: 0: Type of plan unknown (default) 1: ISDN telephony (CCITT E.164) 3: Data numbering plan (CCITT X.121) 4: Telex numbering plan (CCITT F. 69) 5: Reserved for national use 6: Land mobile numbering plan (CCITT E212) 8: National numbering plan 9: Private numbering plan A: Internet, IP-number v4 B: Internet, IP-number v6 Derivation: Optional. From bits 4 - 1 of octet 1 of the GSM Address String type as defined in TS GSM 09.02. The list of values is a comprehensive list of known values and some might not occur in practice. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
A_NUMBER	X(40)	Identifies the billable party; for example, the ISDN number, call-line-identity, or source IP address. For MTC and MOC calls, this number contains the subscriber number to be billed, which has not automatically to be the originating number. Condition: Can be used as an alternative to the IMSI, but could also be an Internet account number. Values: Defined in TS GSM 03.03 or in international notation. Should always be: *International_access_codeCountry_codeNational_destination_codeSubscriber_number* Examples: Fixed: +49410676810 Mobile: 01729183333 (Roaming-MOC: 0000*MNC/MCC*, 000026202) IPv4: 192.168.10.2 (always 4 token, each 3 decimals) IPv6: ABCD:10:2:1AF:0:1F0A:F:1F0 (always 8 token, each 4 hex) Derivation: Mandatory. From the GSM item MSISDN as defined in TS GSM 12.05. Set by the first processor and left unchanged, but is normalized: Fixed: 0049410676810 Mobile: 00491729183333 (Roaming-MOC: 0000*MNC/MCC*, 000026202) IPv4: 192168010002 (dots removed, filled with leading zeros) IPv6: ABCD0010000201AF00001F0A000F01F0 (colons removed, filled with leading zeros)
B_MODIFICATION_INDICATOR	H(2)	See A_MODIFICATION_INDICATOR. Optional.
B_TYPE_OF_NUMBER	Z(1)	See A_TYPE_OF_NUMBER. Mandatory.
B_NUMBERING_PLAN	X(1)	See A_NUMBERING_PLAN. Optional, only if available.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
B_NUMBER	X(40)	Identifies the called number. Condition: If there is no called number available (for example, Internet or Telematic), a dummy number has to be inserted instead; for example, 0049. If the rating does not involve zoning, profile-related features, or special numbers, the BRM rating works with even an incomplete called number. Values: Defined in TS GSM 03.03 or in international notation. Should always be: *International_access_codeCountry_codeNational_destination_codeSubscriber_number* Examples: Fixed: 0049410676810 (normal) Fixed: 0049112 (emergency) Fixed: 004970012345678 (vanity) Fixed: 004911833 (special) Mobile: 00491729183333 (normal) Mobile: 0049172559183333 (mailbox) Mobile: 00490172112 (emergency) Mobile: 0049017222255 (special mobile-number) Mobile: 004911833 (special fixed-number) Mobile: 000026202 (Roaming-MTC: 0000*MNC/MCC*) IPv4: 192.168.10.2 (always 4 token, each 3 decimals) IPv6: ABCD:10:2:1AF:0:1F0A:F:1F0 (always 8 token, each 4 hex) Derivation: Mandatory. From the GSM item Called Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02. Set by the first processor and left unchanged, but is normalized. Fixed: 0049410676810 Mobile: 00491729183333 (Roaming-MOC: 0000*MNC/MCC*, 000026202) IPv4: 192168010002 (dots removed, filled with leading zeros) IPv6: BCD0010000201AF00001F0A000F01F0 (colons removed, filled with leading zeros)
DESCRIPTION	X(50)	Description text for this usage scenario; for example, call destination description for the B Number or service description used. Values: For example, "HAMBURG" or "004940" For example, "Travel Info" or "Wakeup Call" Value is related to the original input format. Derivation: Optional. Calculated by a rating processor or directly taken out of the original CDR stream. Might be changed by any processor.
C_MODIFICATION_INDICATOR	H(2)	See A_MODIFICATION_INDICATOR. Optional. Only mandatory if C Number is present.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
C_TYPE_OF_NUMBER	Z(1)	See A_TYPE_OF_NUMBER. Optional. Only mandatory, if C Number is present.
C_NUMBERING_PLAN	X(1)	See A_NUMBERING_PLAN. Optional. Only mandatory if C Number is present.
C_NUMBER	X(40)	Third-party number; for example, where the call was first terminated in the case of terminated transit or routed, forwarded calls. This field contains the number initiating the call forwarding. **Note:** To avoid any doubt, the 'A to C' and 'C to B' legs of a call-forwarding scenario must be treated as separate calls and the originating and terminating records should never be chained together. Condition: Only present where it is available. Values: See B_NUMBER. Derivation: Optional. From the GSM item Called Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02. Set by the first processor and left unchanged.
USAGE_DIRECTION	X(1)	Describes the direction of the connection that was established. Can be used by any rating and post-processor to identify the direction of the event; for example, to determine a specific call scenario. Values: 0: Originated usage 1: Terminated usage 2: Roaming originated usage 3: Roaming terminated usage Examples: 0: MOC, mobile originated WAP, … 1: MTC, mobile terminated WAP, … 2: roaming MOC 3: roaming MOC Derivation: Mandatory. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CONNECT_TYPE	X(2)	Type of connection. Can be used to identify the type of the call; for example, to determine a specific call scenario. Values: 01: Mobile to Mobile 02: Mobile to Land 03: Land to Mobile 04: Land to Land 10: Effective POTS Call 11: Effective ISDN Call 12: Effective Call Diversion 13: Subscriber Procedure 14: Subscriber Service Command 15: Effective ISDN-E Call 16: Ineffective POTS Call 17: Ineffective ISDN Call 18: Ineffective Call Diversion 19: Ineffective ISDN-E Call 20: Non Call Related Output 30: Anonymous login Other values might apply according to the related original input format. Derivation: Mandatory. Set by the first processor and left unchanged.
CONNECT_SUB_TYPE	X(2)	Detailed description of the connection or call type. Can be used to identify the type of the call; for example, to determine a specific call scenario. Values: 01: Mobile-Call 02: Local-Call (for example, BestCity, BestFriend, etc.) 03: National-Call 04: International-Call Derivation: Mandatory. Set by the first processor and left unchanged.
CALLED_COUNTRY_CODE	String	Derivation: Optional, but should be set when the CONNECT_TYPE indicates an international call
BASIC_SERVICE	X(3)	Uniquely identifies the basic usage-related service. A service is defined by: • Service type • Service code

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
BASIC_SERVICE	X(1)	Specifies the type of basic service. Values: 0: Tele Service (for example, GSM Tele, ISDN, analog, standard, etc.) 1: Bearer Service 2: Supplementary Service 3: Telematic Service (only if present) 4: Internet Service 5: ISDN mobile (only if present) 6: Mailbox (only if present) 7: VPN mobile (only if present) 8: GPRS 9: Switch related (for example, for direct fixed network support) E: Event/VAS Services W: WAP Other service types might be defined when necessary. Derivation: Mandatory. Set by the first processor and left unchanged.
SERVICE_CODE	X(2)	The Service Code is either a Teleservice Code or Bearer Service Code as determined by Service Type. Values: **Note:** Other values might apply according to the related original input format.All Tele Services: 10: All Voice 11: Telephony (default) 12: Emergency calls 13: GPRS (default) 14: HSCSD (default) 15: WAP (default) 20: All Short Message Service 21: Short Message MT/PP 22: Short Message MO/PP 30: MHS 31: Advanced MHS access 40: All Videotext 41: Videotext access profile 42: Videotext access profile 2 43: Videotext access profile 3 50: All Teletext Transmission Services 51: Teletext (Circuit Switch) 52: Teletext (Packet Switch) 61: Facsimile Group 3 & alternative voice 62: Automatic Facsimile Group 3 63: Automatic Facsimile Group 4

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SERVICE_CODE (continued)	X(2)	All Bearer Services: 10: 3.1 KHz Bearer Group 20: Circuit data asynchronous 21: Duplex Asynchronous 300bps data circuit 22: Duplex Asynchronous 1200bps data circuit 23: Duplex Asynchronous 1200/75bps data circuit 24: Duplex Asynchronous 2400bps data circuit 25: Duplex Asynchronous 4800bps data circuit 26: Duplex Asynchronous 9600bps data circuit 30: Circuit data synchronous 32: Duplex Synchronous 1200bps data circuit 34: Duplex Synchronous 2400bps data circuit 35: Duplex Synchronous 4800bps data circuit 36: Duplex Synchronous 9600bps data circuit 40: PAD access asynchronous 41: Duplex Asynchronous 300bps PAD access 42: Duplex Asynchronous 1200bps PAD access 43: Duplex Asynchronous 1200/75bps PAD access 44: Duplex Asynchronous 2400bps PAD access 45: Duplex Asynchronous 4800bps PAD access 46: Duplex Asynchronous 9600bps PAD access 50: PAD access synchronous 54: Duplex Synch. 2400bps PAD access 55: Duplex Synch. 4800bps PAD access 56: Duplex Synch. 9600bps PAD access 60: All alternate voice/data c.d.a 61: Alt. Voice/Asynch. 300bps unrest'd digital 62: Alt. Voice/Asynch. 1200bps unrest'd digital 63: Alt. Voice/Asynch. 1200/75bps unrest'd digital 64: Alt. Voice/Asynch. 2400bps unrest'd digital 65: Alt. Voice/Asynch. 4800bps unrest'd digital 66: Alt. Voice/Asynch. 9600bps unrest'd digital 70: Alternate voice data c.d.s 72: Alt. Voice/Synch. 1200bps unrest'd digital 74: Alt. Voice/Synch. 2400bps unrest'd digital 75: Alt. Voice/Synch. 4800bps unrest'd digital 76: Alt. Voice/Synch. 9600bps unrest'd digital 80: Voice followed by data c.d.a

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
SERVICE_CODE (continued)	X(2)	All Bearer Services: (continued) 81: Voice then Asynch. 300bps unrest'd digital 82: Voice then Asynch. 1200bps unrest'd digital 83: Voice then Asynch. 1200/75bps unrest'd digital 84: Voice then Asynch. 2400bps unrest'd digital 85: Voice then Asynch. 4800bps unrest'd digital 86: Voice then Asynch. 9600bps unrest'd digital 90: All Voice followed by data c.d.s 92: Voice then Synch. 1200bps unrest'd digital 94: Voice then Synch. 2400bps unrest'd digital 95: Voice then Synch. 4800bps unrest'd digital 96: Voice then Synch. 9600bps unrest'd digital
SERVICE_CODE (continued)	X(2)	All Telematic Service: 01: SMS 02: E-Mail 03: Pull-Service 04: Short-Fax 05: Push-Service All Internet Service: 10: all Internet 11: direct Access 12: WebMaster Emergency 13: Voice over IP 14: Fax over IP 20: E-Mail 30: Active Channel 40: Video on demand 41: Music on demand 50: Newsgroup access 62: Internet Fax other values Can be used according to the network. All Switch-related Services or WAP/GPRS Services: 01...99: see related switch documentation (values used 1:1) AA...ZZ: see related switch documentation (values used 1:1) All Event/VAS Services: 00: all default Event/VAS usage other Service Codes might be defined when necessary. Derivation: Mandatory. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
QOS_REQUESTED	X(5)	The type of QoS requested by the Terminal Equipment (TE) at usage setup or the QoS requested to the Network Equipment (NE). The QoS Requested is typically a normalized billable QoS value. For detailed network-related QoS attributes, see the related Associated Service Extension Record. Condition: The use of a QoS might not be appropriate; for example, call forwarding cases, short message services. Applies only where appropriate information is available. Values for Radio Channel: H(2) 00: Half Rate Channel 01: Full Rate Channel 02: Dual Rate Channel, Half Rate Preferred 03: Dual Rate Channel, Full Rate Preferred Values for Quality of Service: X(5) xxxxx: Any alphanumeric representation of the NE Value is according to the related original input format. Derivation: Optional. From the GSM item RadioChanRequested, a component of RadioChanInfo as defined in TS GSM 12.05. or directly out of the NE-interface. Encoded as per TS GSM 04.08. Set by the first processor and left unchanged.
QOS_USED	X(5)	The type of QoS negotiated by the network. The QoS used is typically a normalized billable QoS value. For detailed network-related QoS attributes, see the related Associated Service Extension Record. Condition: The use of a QoS might not be appropriate; for example, call forwarding cases, short message services. Applies only where appropriate information is available. Values for Radio Channel: H(2) 00: Half Rate Channel 01: Full Rate Channel Values for Quality of Service: X(5) xxxxx: any alphanumeric representation of the NE Values is according to the related original input format. Derivation: Optional. From the GSM item channel type, a component of RadioChannel-Used as defined in TS GSM 12.05. Encoded as per TS GSM 04.08 or directly out of the NE-interface. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CALL_COMPLETION_INDICATOR	X(3)	Indicates whether a call was correctly completed or not. Optionally defines a close cause reason code. Single-Byte Values X(1): if no reason code is available: C: Call completed, charged as usual D: Call dropped, treatment open, but will first be charged as usual T: Call completed, test call, will not be charged Optional Values after rating processor: 0: After rating processor: not rated yet: should be rated later by the billing post-processor: call completed 1: After rating processor: rated: should not be rated again by a billing post-processor - call completed 2: After rating processor: not rated yet: should be rated later by the billing post-processor: call dropped 3: After rating processor: rated: should not be rated again by a billing post-processor: call dropped Double-Byte Values X(2) - if a reason code is available: 00: normal 01: partial record 02: partial call re-establishment 03: unsuccessful call attempt 04: abnormal release 05: camel init call release 16: volume limit 17: time limit 18: network element switch 19: max. change condition 20: management intervention other values can be used according to the network. Derivation: Mandatory, default C. Set by the first processor and left unchanged.
LONG_DURATION_INDICATOR	X(1)	Specifies which part of the call, in the case of split calls. Values: S: Single (only one record present) F: First (the first record in the row of split records) I: Intermediate (one of the middle records in the row of records) L: Last (the last record in the row of split records) Derivation: Mandatory, default S. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
CHARGING_START_TIMESTAMP	YYYYMMDDHHMISS	Timestamp used for start of charging. In the mobile originated case, this is as determined by the VPLMN's charging rules. In the mobile terminated case, it is also at the discretion of the VPLMN, even though the information is for use in charging by the VPLMN. Format: YYYYMMDDHHMISS; for example, 19990518190357 Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory. From the GSM item answer-time or seizure-time as defined in TS GSM 12.05. Set by the first processor and left unchanged.
CHARGING_END_TIMESTAMP	YYYYMMDDHHMISS	Timestamp used for end of charging. In the mobile originated case, this is as determined by the VPLMN's charging rules. In the mobile terminated case, it is also at the discretion of the VPLMN, even though the information is for use in charging by the VPLMN. Format: YYYYMMDDHHMISS; for example, 19990518190357 Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Optional. Might be set by the first processor and left unchanged. **Note:** If not present, this value can be calculated by using the start timestamp and the duration.
CREATED_TIMESTAMP	Date	Optional. The time that the event was created in BRM.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
UTC_TIME_OFFSET	X(5)+/-HHMI	All timestamps are VPLMN or originating network local time. So that the time can be equated to time in the HPLMN or recipient network, the sender gives the difference between local time and UTC time. Can be used to translate the CHARGING_START/END_TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Values: UTC Time Offset = Local Time minus UTC Time Example: Washington DC, USA 1000hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 10 - 15 = -0500 Madrid, Spain 1600hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 16 - 15 = +0100 Sydney, Australia 0100hrs 11/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = (01 + 24) - 15 = +1000 (Note that where dates are different, 24 is added to the time of the greater date.) Derivation: Mandatory. Set by the first processor and left unchanged.
DURATION	9(15)	Duration-based charge indicates that the field represents a Chargeable Duration. Can be used to evaluate all duration-based functions; for example, determination of the pricing rating steps. Condition: For event-based charges or an inter-network account charge, the field is not relevant. URC.01 implementation of the TD.17 item Chargeable Units. Maximum Value: 999999999999999 Derivation: Mandatory, default 0. Set by the first processor and left unchanged but might be patched by some kind of rating processors.
TOTAL_CALL_EVENT_DURATION	Integer	The total duration of the event. This should be set for all time-based services; for example, telephony. Mandatory. The default value is **0**.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
DURATION_UoM	X(3)	Unit of Measurement associated with Chargeable Quantity Value. Can be used to interpret the quantity value, but usually not needed, because the quantity itself is sufficient for all rating steps. Values: SEC: Seconds (default) MIN: Minutes HRS: Hours or any other applicable value describing a timed quantity unit of measurement. Derivation: Mandatory, default 'SEC'. Set by the first processor and left unchanged.
VOLUME_SENT_	9(15)	In addition to the basic duration quantity value, a special volume might be defined to keep an additional rating relevant measurement. This is typically BYTES sent by the initiator (A number). Maximum Value: 999999999999999 Can be used to evaluate additional volume-based functions; for example, determination of the pricing rating steps. Derivation: Mandatory, default 0. Set by the first processor and left unchanged but might be patched by some kind of rating processors.
VOLUME_SENT_UoM	X(3)	The Unit of Measurement associated with VOLUME_SENT. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: BYT: Bytes/Characters (default) KBY: Kilobytes MBY: Megabytes GBY: Gigabyte or any other applicable value describing a metered quantity unit of measurement. Derivation: Mandatory, default 'BYT'. Set by the first processor and left unchanged.
VOLUME_RECEIVED	9(15)^	In addition to the basic duration value, a special volume might be defined to keep an additional rating relevant measurement. This is typically BYTES received by the initiator (A Number). Maximum Value: 999999999999999 Can be used to evaluate an additional volume-based functions; for example, determination of the pricing rating steps. Derivation: Mandatory, default 0. Set by the first processor and left unchanged but might be patched by some kind of rating processors.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
VOLUME_RECEIVED_UoM	X(3)	The Unit of Measurement associated with VOLUME_RECEIVED value. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: • BYT: Bytes/Characters (default) • KBY: Kilobytes • MBY: Megabytes • GBY: Gigabyte • Any other applicable value describing a metered quantity unit of measurement Derivation: Mandatory, default 'BYT'. Set by the first processor and left unchanged.
NUMBER_OF_UNITS	9(15)	One of the following: • Original charged units (for example, beats, clicks) as applied by the sender • Rounded total volume charged by the sender • Number of events associated with this record (for example, number of SMS messages or number of internet hits/clicks) Might be useful for analyzing how many units the event was originally treated by or for storing a fourth quantity. Condition: Applies only if available. Alternative URC.01 implementation of the TD.17 item Chargeable Units. Maximum Value: 4294967296 Derivation: Mandatory, default 0. Set by the first processor and left unchanged, but might be changed by some type of rating processors.
NUMBER_OF_UNITS_UoM	X(3)	The Unit of Measurement associated with NUMBER_OF_UNITS value. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: CLK: Clicks (anonymous quantity) (default) MSG: Messages PAG: Pages PAC: Packets PIC: Pieces RTS: Points MTR: Meters KMR: Kilometer SPD: Speed TRN: Transactions or any other applicable value describing a metered quantity unit of measurement. Derivation: Mandatory, default CLK. Set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
RETAIL_IMPACT_CATEGORY	X(10)	Impact category defining the usage scenario specific rate; for example, the zone value used for customer rating. Values: 00000: undefined impact category (default) 00001 - 99999: user defined Alternatively a user-defined string can be used as a value. Derivation: Optional, but mandatory after any rating processor, default 00000. Might be changed by any processor.
RETAIL_CHARGED_AMOUNT_VALUE	9(11)	The charge for the event (for example, the retail price). This includes any toll charge but does not include any CAMEL invocation fee. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Optional, but mandatory after any customer rating processor.
RETAIL_CHARGED_AMOUNT_CURRENCY	X(3)	Currency code as defined within the associated charge; for example, DEM or EUR. Derivation: Optional, but mandatory whenever the RETAIL_CHARGED_AMOUNT_VALUE is set. Use the three-digit ISO currency code.
RETAIL_CHARGED_TAX_TREATMENT	X(1)	Charges might be inclusive or exclusive of tax. Can be used to interpret the amount value and to distinguish between net and gross charges. Values: Y: Tax included in the charge N: Tax not included in the charge (default) Derivation: Optional, but mandatory whenever the RETAIL_CHARGED_AMOUNT_VALUE is set.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
RETAIL_CHARGED_TAX_RATE	9(4)	Defines the tax rate applicable to the charge. Because some national legal definitions dictate that the tax rate applicable is determined by the invoice date, there is a possibility that the rate on the invoice might differ from the rate on the transfer. However, the likelihood of this happening is extremely low. Can be used to interpret the amount value and to convert between net and gross charges. Values: 0000 through 9999 (2 fixed decimals) Example: 16.00% 1600 Derivation: Optional, but mandatory whenever the RETAIL_CHARGED_AMOUNT_VALUE is set.
RETAIL_CHARGED_TAX_VALUE	Decimal	Derivation: Calculated, default = 0.0
WHOLESALE_IMPACT_CATEGORY	X(10)	Wholesale/Advice of Charge - Impact category used for purchase rating. Values: 00000: undefined impact category (default) 00001 - 99999: user-defined Derivation: Optional. Might be changed by any processor.
WHOLESALE_CHARGED_AMOUNT_VALUE	9(11)	Wholesale/Advice of Charge: charge for the event (for example, the wholesale price). This includes any toll charges. Can be used to keep the original purchase charge, to evaluate a record-based margin together with the charged amount value. Values: Space: No price given, like NULL in a database Floating point format: Given value, might be 0.000. If no floating point exists, the last three digits are always taken as decimals) Minimum: -9999999999 Maximum: 99999999999 Examples: '00000012500' for 12,50 '-0001200100' for -1.200,10' '00000012.50' for 12,50 '00012.50000' for 12,50 Derivation: Optional. Usually transmitted by the sender (origin network operator) of the file, but might also be recalculated by any processor to represent the purchase charge.
WHOLESALE_CHARGED_AMOUNT_CURRENCY	X(3)	See RETAIL_CHARGED_AMOUNT_CURRENCY.
ZONE_DESCRIPTION	String	Calculated. Used by zoning and rating modules.
IC_DESCRIPTION	String	Used by the zoning and rating modules.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
ZONE_ENTRY_NAME	String	Calculated. Used by zoning and rating modules.
WHOLESALE_CHARGED_TAX_TREATMENT	X(1)	See RETAIL_CHARGED_TAX_TREATMENT.
WHOLESALE_CHARGED_TAX_RATE	9(4)	See RETAIL_CHARGED_TAX_RATE.
WHOLESALE_CHARGED_TAX_VALUE	Decimal	Derivation: Calculated, default = 0.0
TARIFF_CLASS	X(10)	Tariff Class contains tariff information as represented within the original CDR format; for example, wholesale tariff model identification. Can be used to evaluate the original charge configuration in conjunction with the BASIC_AoC_AMOUNT_VALUE. Condition: Only present if original purchase charge information is available. Values: Dependent on the original format. No format conversion will take place. See the appropriate documentation of the original format. Derivation: Optional (but should by mandatory for all mobile (GSM) related records). Might be set by any processor.
TARIFF_SUB_CLASS	X(10)	Contains detailed tariff information as represented within the original CDR format; for example, wholesale zone identification. Can be used to evaluate the origin charge configuration in conjunction with the WHOLESALE_CHARGED_AMOUNT_VALUE. Condition: Only present if origin purchase charge information is available. Values: Dependent on the original format. No format conversion will take place. See the appropriate documentation of the original format. Derivation: Optional (but should by mandatory for all mobile (GSM) related records). Might be set by any processor.
USAGE_CLASS	X(5)	Specifies a format-related usage scenario; for example, call forwarding, roaming, mailbox request, or local calls. Can be used to evaluate the origin call scenario. The call class can be used to convert a scenario into a combined zone value or to identify specific rating specialties. Therefore the call class consists of original record fields. Values: Dependent on the original format. No format conversion or normalization will take place. The content is derived from any rule-based translations of any available raw event attributes, to represent all possible usage scenarios of the origin format. 00000: undefined usage class Derivation: Mandatory. Should be set by the first processor and left unchanged.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
USAGE_TYPE	X(5)	Specifies a customer-related usage scenario; for example, customer-to-customer call, birthday call, or closed-user-group calls. Can be used to evaluate an A Number-customer and B Number-customer related scenario (using direct access to specific customer-info-fields). The call type can be used to convert a scenario into a combined zone value or to calculate a record-based discount when estimating the charging amounts. Values: User definable values might be used. The content of the field depends on the rule-based configuration. 00000: undefined usage type Derivation: Optional. Might be changed by any rating or billing processor.
EVENT_TYPE	String	BRM event type.
SERVICE_TYPE	String	BRM service type. When the service has a subscription service, the string is separated by semicolons; for example, /service/gsm/data;/service/gsm)
BILLCYCLE_PERIOD	YYYYMMBC	Defines the next open billcycle period this event belongs to. Can be used to group or split the EDR stream into billcycle-related smaller portions. Condition: Only present if a rating or pre-billing processor evaluates the next billcycle period for the A number customer. Values: YYYY: The actual year of the next open billcycle period. MM: The actual month of the next open billcycle period. BC: The billcycle identifier. Derivation: Optional, but should be mandatory for any pre-billing processor.
PREPAID_INDICATOR	9(2)	Specifies if the event is a prepaid event. Can be used to identify prepaid scenarios within a mixed post-/prepaid environment. Values: Default: no prepaid scenario prepaid scenario Derivation: Mandatory. Set by the first processor and left unchanged.
NUMBER_ASSOCIATED_RECORDS	9(2)	Number of associated records attached to this basic detail record. Can be used to evaluate how many associated records have to be read ahead. Values: 00: No associated records attached, next record is a basic one. 01-99: Number of associated records followed by this record. Derivation: Mandatory. Must be changed by any processor if new associated records are added.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
NUMBER_OF_CDRS	Integer	Number of CDRs that were compiled into this EDR during call assembly. Derivation: Optional. Calculated.
ERROR_REJECT_TYPE	String	Used by the FCT_Reject to reject the DETAIL to another stream than the standard reject stream. Derivation: Optional, default = ' '
OPERATOR_SPECIFIC_INFO	String	Stores a key to identify the CDR used to generate a specific EDR. Useful for RAP or CIBER return. Derivation: Optional, default = ' ' Must be set by an iScript.
DISCOUNT_KEY	String	N/A
GEOGRAPHICAL_LOCATION	String	Conditional in TAP 3.10. Indicates the geographical location of the terminal equipment. Format: This field contains comma-separated tag-value pairs that indicate the geographical location of the serving network, serving BID, serving location description, longitude, and latitude. The tag values of the corresponding fields are as follows: • ServingNetwork: 1 • ServingBID: 2 • ServingLocationDescription: 3 • Longitude: 4 • Latitude: 5 Example 1: If the TAP field values are as follows: • ServingNetwork: **AIRTEL** • ServingBID: **AIRBID** • ServingLocationDescription: **Bangalore** • Longitude: **111** • Latitude: **103** The value of DETAIL.GEOGRAPHICAL_LOCATION would be: **1,AIRTEL, 2,AIRBID, 3,Bangalore, 4,111,5,103** Example 2: If the TAP field values are as follows: • ServingNetwork: **AIRTEL** • ServingBID: **AIRBID** • Latitude: **103** The value of DETAIL.GEOGRAPHICAL_LOCATION would be: **1,AIRTEL, 2,AIRBID, 5,103**
FRAUD_MONITOR_INDICATOR	String	Conditional in TAP 3.10. Indicates that the chargeable subscriber is flagged for fraud information collection purposes. Possible values: **1 -** Fraud Monitored Subscriber If the field is present, it should have a value of **1**.
ORIGINAL_BATCH_ID	String	Optional, but might be set equal to the original file batch ID.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
BATCH_ID	String	Optional, but might be set equal to the recycle or rerate file batch ID.
NE_CHARGING_START_TIMESTAMP	Date	Network Element date/time stamp. Time at which the call started. Derivation: Optional.
NE_CHARGING_END_TIMESTAMP	Date	Network Element date/time stamp. Time at which the call ended. Derivation: Optional.
UTC_NE_START_TIME_OFFSET	X(5)	Optional.
UTC_NE_END_TIME_OFFSET	X(5)	Optional.
UTC_END_TIME_OFFSET	X(5)	Timezone where the call terminated. Derivation: Optional.
INCOMING_ROUTE	X(30)	Incoming route name. Optional.
ROUTING_CATEGORY	X(20)	Category denoting the routing of the call to the destination party. Optional.
DISCARD_REASON	String	The reason for discarding the EDR. This field is set by FCT_Discard.
CREDIT_LIMIT_CHECK	Integer	Specifies whether to perform credit limit checking on the EDR: • 1 = Perform a credit limit check • 0 = Skip the credit limit check Mandatory.
CREDIT_LIMIT_CHECK_RESULT	Integer	Specifies whether the EDR passed or failed the credit limit check: • 1 = The EDR passed the simple credit limit check • 0 = The EDR failed the simple credit limit check Mandatory.
UNRATED_QUANTITY	Decimal	Unrated quantity filled in after credit limit check.
REFRESH_BALANCE	Integer	Specifies whether the latest balance information should be retrieved from the database. When this field is set, the discounting module calls the balance module to get the latest balance information from the database, whether or not a balance packet is present in the EDR.
OBJECT_CACHE_TYPE	Integer	Cache residency type. • **0**: Convergent • **1**: Prepaid • **2**: Postpaid
DELAYED_ERROR_BLOCK	String	Stores the block name that has the fatal error.
EVENT_ID	String	Used by Revenue Assurance.
ITEM_TAG	String	Used by FCT_ItemAssign. Calculated.
RERATE_TAG	Integer	Used for re-rating

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
DROPPED_CALL_QUANTITY	Decimal	Duration of a dropped call.
DROPPED_CALL_STATUS	Integer	Status of a dropped call. • 0 = No dropped call service-level ERA associated with the service. • 1 = The call is a dropped call. • 2 = Continuation call. • 3 = Both a dropped call and a continuation call. • 4 = Does not meet the criteria for either a dropped call or a continuation call.
NET_QUANTITY	Decimal	Contains the summation of the BALANCE_PACKET.PIN_QUANTITY for the associated RUM.
INTERN_attributes (shown below)	As shown below	The following INTERN_ attributes are used by specific modules to temporarily store calculated values. They are all mandatory, but only from a definition point of view. The content value of these fields will be filled automatically by the appropriate modules. All other modules should use these values as read only.
INTERN_ZONE_MODEL	Integer	The internal zone model used by zoning, rating, and discounting Mandatory. Calculated.
INTERN_NETWORK_MODEL	String	The internal network model code. Mandatory. Calculated.
INTERN_NETWORK_OPERATOR	String	The internal network operator code. Used by Interconnect aggregation. Mandatory. Calculated.
INTERN_APN_GROUP	String	The internal APN group code used by zoning. Mandatory. Calculated.
INTERN_TERMINATING_SWITCH_IDENTIFICATION	String	The internal terminating switch ID. Used by output mapping Mandatory. Calculated.
INTERN_BILLING_CURRENCY	String	The internal billing currency used by exchange rate conversion. Mandatory. Calculated.
INTERN_HOME_CURRENCY	String	The internal home currency used by exchange rate conversion. Mandatory. Calculated.
INTERN_SLA_USC_GROUP	String	The internal customer-related SLA-based usage scenario map group code. Mandatory. Calculated.
INTERN_SLA_RSC_GROUP	String	The internal customer-related SLA-based rate service class map group code Mandatory. Calculated.
INTERN_SLA_IRULE_SET	String	The internal customer-related SLA-based irule_set-code. Mandatory. Calculated.
INTERN_PROCESS_STATUS	Integer	Possible values are • 0 = normal (default) • 1 = recycling • 2 = recycling-test Mandatory. Calculated.

Table 79-6 (Cont.) Basic Detail Record Fields

Name	Format	Description
INTERN_BALANCE_GROUP_ID	String	The balance group of the service to which the event belongs. Optional.
INTERN_SERVICE_BILL_INFO_ID	String	The billinfo of the service's balance group. Optional.
INTERN_DISCOUNT_OWNER_ACCT_ID	String	Optional.
INTERN_SERVICE_BILL_INFO_ID	String	The billinfo of the service's balance group. Optional.
ACCOUNT_ID	String	The POID of the Customer A account. Optional.
TB_RECORD_NUMBER	Integer	The Trigger Billing Record number used by the Trigger Bill Output Mapping to assign the array index set by the grammar. Optional.
RAP_FILE_SEQ_NO	String	Indicates the Returned Account Procedure (RAP) file in which the Recipient PMN returned the TAP file record to the Sender PMN. This field is a unique reference. Used in TAP files. Optional.
PROFILE_LABEL_LIST	String	A list of unique labels of all shared profiles having attributes matching a specific EDR field or event attribute. Optional. Calculated.
DROPPED_CALL_QUANTITY	Decimal	When the EDR is flagged as a continuation call, this field stores the duration of the associated dropped call.
DROPPED_CALL_STATUS	Integer	Specifies whether the EDR is for a normal call (**0**), a dropped call (**1**), a continuation call (**2**), both a dropped call and a continuation call (**3**), or an already processed EDR (**4**).

Associated Revenue Assurance Extension Record

Table 79-7 lists the fields in the Associated Revenue Assurance Extension Record. This record is optional with an occurrence of 0 or 1 time only.

Table 79-7 Associated Revenue Assurance Extension Record Fields

Name	Format
BATCH_ID	String
CDR_FILE_NAME	String
START_TIME	String
EDR_STATUS	String
REVENUE_STREAM	String
OUTPUT_STREAM	String
DISCOUNT_AMOUNT	Decimal
OLD_DISCOUNT_AMOUNT	Decimal

Table 79-7 (Cont.) Associated Revenue Assurance Extension Record Fields

Name	Format
CHARGED_AMOUNT	Decimal
OLD_CHARGED_AMOUNT	Decimal

Associated GSM/Wireline Extension Record (RECType 520)

This record is optional and will be generated only if the related Basic Detail Record indicates a GSM or Wireline service. Table 79-8 describes the fields in the Associated GSM/Wireline Extension Record.

Table 79-8 Associated GSM/Wireline Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 520 - GSM/Wireline Extension Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PORT_NUMBER	X(24)	Identifies the customer to charge; for example, the IMSI or SIM number. Condition: For Value Added Services and APLMN Service Center Usage, either the IMSI or MSISDN might be supplied, although one of them must be supplied and, where available, the IMSI is preferred. For normal mobile calls, the SIM number is preferred. Derivation: Optional. From the GSM item served IMSI as defined in TS GSM 12.05. Defined in TS GSM 03.03.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
DEVICE_NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Condition: Not present where the terminal equipment is not involved in the call; for example, in forwarded call cases. It is not mandatory for the VPLMN to transfer this information. Derivation: Optional. From the GSM item IMEI as defined in TS GSM 12.05. Defined in TS GSM 03.03. **Note:** Even though the IMEI is 16 digits in length, the check digit is not transmitted.
A_NUMBER_USER	X(40)	The customer who owns the number from which the call was originated, for terminated calls. Not used for rating, but could be used on invoices. Condition: There is no calling number present where it is unavailable. Could be different from the A Number; for example, in case of VPN calls. For VPN calls, the A Number contains the party to be billed, and this field contains the user initiating the call. Values: See A_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
DIALED_DIGITS	X(40)	The number dialed by the customer when establishing a call, or the number to which the call is forwarded or transferred. Can be used for managing disputes. Condition: There might be no called number for the basic service emergency call but operators might optionally insert the digits 112 or their national emergency number into this field. The notation should always be local; for example, 04106768124. Values: See B_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan, and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
BASIC_DUAL_SERVICE	X(3)	A dual service can be used in context with twin or duo cards.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
VAS/PRODUCT_CODE	X(10)	A classification of Value Added Services as generated by the sender. Can be used to map to a specific internal service code to implement specific usage scenarios for any rating purposes. Values: VMAIL: Voice Mail Services SEC: Secretarial Services OPER: Telephonic Operator Services FI: Financial Information TRAVEL: Travel Information This is not a definitive list and might be added to through MoU-TADIG from time to time or might be user defined. Derivation: Optional. From the GSM item vasCode as defined in GSM TD17. Set by the first processor and left unchanged.
ORIGINATING_SWITCH_IDENTIFICATION	X(15)	Identifies the MSC or SwitchID handling the origin of the call. In case of mobile roaming calls (GSM), this field contains the MOC-related MCC/MNC. In case of wireline networks, this field contains the primary switch that generated this CDR. Can be used by any interconnect rating processor to uniquely identify the trunk names but will only be used if the trunk names are only unique within the related switch. See TRUNK_INPUT and TRUNK_OUTPUT. Can also be used to normalize the A Number for MOC roaming. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: • The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. • The signaling point code as per GSM 03.03; for example, 253464. • The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201. • MCC = Mobile Country Code, MNC = Mobile Network Code • A name; for example, "HELSINKI": Must be uppercase. • A switchID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case of interconnect records (for intercarrier rating/billing reasons,) or for mobile (roaming) records. **Note:** The switch might not be needed if the trunk names are unique within the total network). Set by the first processor and left unchanged.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
TERMINATING_SWITCH_IDENTIFICATION	X(15)	Identifies the MSC or Switch ID handling the termination of the call. In case of mobile roaming calls (GSM), this field contains the MTC-related MCC/MNC. In case of wireline networks, this field contains the secondary switch or is empty. Can be used to normalize the B Number in case of MTC-roaming cause. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: • The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. • The signaling point code as per GSM 03.03; for example, 253464. • The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201 • MCC = Mobile Country Code; MNC = Mobile Network Code • A name; for example, "HELSINKI": Must be uppercase. • A switch ID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case mobile (roaming) records. Set by the first processor and left unchanged.
TRUNK_INPUT	X(15)	Trunk identification, inroute address in network switches. Used for interconnect rating to identify the inroute leg of a call. The inroute leg references a related network operator from which the call was received and how to treat this inroute leg in case of intercarrier rating. Encoding: Must uniquely identify a bundled line trunk: • Within the given ORIGINATING_SWITCH_IDENTIFICATION. • With the global network structure. Derivation: Optional, only mandatory in case of interconnect records (for intercarrier rating/billing reasons). Set by the first processor and left unchanged.
TRUNK_OUTPUT	X(15)	Trunk identification, outroute address in network switches. Can be used by any interconnect rating processor to identify the outroute leg of a call. The outroute leg references a related network operator to which the call was routed or terminated and how to treat this outroute leg in case of intercarrier rating. Encoding: Must uniquely identify a bundled line trunk: • Within the given TERMINATING_SWITCH_IDENTIFICATION. • With the global network structure. Derivation: Optional, only mandatory in case of interconnect records. Set by the first processor and left unchanged.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
LOCATION_AREA_INDICATOR	X(10)	Identifies the MSC responsible for handling the call and the location of the equipment making or receiving the call. The definition of these items can be found in the Data Dictionary under MSC Identification, Location Area, and Cell Id. Can be used to map to a specific internal service code to implement a event-dependent rating. Condition: Is not available if not supported by the network or the call does not terminate at the equipment; for example, in call forwarding cases. Values: The Location Area Code is a two-octet string as defined in TS GSM 04.08. For the TAP, the octets are converted to a decimal number in the range 0x00000000 to 0xFFFFFFFF. Derivation: Optional. From the GSM item locationAreaCode as defined in TS GSM 12.05 or directly taken from the sender (VAS). Set by the first processor and left unchanged.
CELL_ID	X(10)	The cell from which the call originated. Can be used to identify the location of the caller. Condition: Operators might not transfer the cell identity. Only available if the call originates or terminates from a mobile phone; for example, not available in call divert cases. Values: The cell identity is a two-octet string as defined in TS GSM 04.08. However, an original hex value is copied. Derivation: Optional. From the GSM item Cell Id as defined in TS GSM 12.05. Set by the first processor and left unchanged.
MS_CLASS_MARK	9(1)	The power capability of the equipment making or receiving the call. Mobiles and transmobiles usually have class 2 capability, handhelds class 4, and PCN applications class 5. Some transmobiles have reduced capability and are classified as class 3. Usually not used. Condition: Only available if supported by the network and the call originates or terminates from the equipment. Is not available in call forwarding cases. Values: 1. Class Mark 2 2. Class Mark 3 3. Class Mark 4 4. Class Mark 5 Other values might apply according to the related original input format. Derivation: Optional. From the GSM item msclassmark as defined in TS GSM 12.05. Set by the first processor and left unchanged.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
TIME_BEFORE_ANSWER	9(5)	The number of seconds until a call was successfully established, defined by the time between the call setup attempt and call answer. Can be used as a QoS parameter. Values: Minimum: 00000 Maximum: 99999 Derivation: Optional. Set by the first processor and left unchanged.
BASIC_AoC_AMOUNT_VALUE	9(11)	A monetary amount assigned to the event by any rating processor and charged to the recipient of the file. This does not include any surcharges. Used for roaming or interconnect rating. Can be used to keep the original purchase charge, to evaluate a record based margin together with the charged amount value. Values: Space: No price given, like NULL in a database Floating point format: Given value, might be 0.000. If no floating point exists, the last three digits are always taken as decimals) Minimum: -9999999999 Maximum: 99999999999 Example: '00000012500' for 12,50 '-0001200100' for -1.200,10 '00000012.50' for 12,50 '00012.50000' for 12,50 Derivation: Optional. Usually handed over by the sender of the file, but might also be recalculated by any processor to represent the purchase charge.
BASIC_AoC_AMOUNT_CURRENCY	X(3)	See RETAIL_CHARGED_AMOUNT_CURRENCY.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
ROAMER_AoC_AMOUNT_VALUE	9(11)	A monetary amount assigned to the event by any rating processor and charged to the recipient of the file. This is typically a special add-on or surcharge. **Note:** The total wholesale charge of a roaming event should be calculated as: Basic AoC Amount + Roamer AoC Amount Used for roaming and interconnect rating. Can be used to keep the original purchase charge, to evaluate a record based margin together with the charged amount value. Values: Space: No price given, like NULL in a database Floating point format: Given value, might be 0.000. If no floating point exists the last 3 digits are always taken as decimals) Minimum: -999999999 Maximum: 99999999999 Example: '00000012500' for 12,50 '-0001200100' for -1.200,10 '00000012.50' for 12,50 '00012.50000' for 12,50 Derivation: Optional. Usually handed over by the sender (origin network operator) of the file, but might also be recalculated by any processor to represent the purchase charge.
ROAMER_AoC_AMOUNT_CURRENCY	X(3)	See RETAIL_CHARGED_AMOUNT_CURRENCY.
NUMBER_OF_SUPPLEMENTARY_SERVICE_PACKETS	9(2)	Defines the number of Supplementary Service Records following these base fields. For example, 05 means that 5 records are following. Can be used to evaluate how the record structure continues. Values: 00 - 99: Either zero or N records are following Derivation: Mandatory. Dependent on the input how many supplementary service records are present.
NUMBER_OF_BS_PACKETS	Integer	Defines the number of Basic Service Records following the Supplementary Service Record. Values: Default = 0 Derivation: Mandatory. Dependent on the number of basic service records present.
SERVING_NETWORK	String	Conditional in TAP 3.10 files. Indicates the network in which the call event was originally created. This field is a unique identifier.
B_CELL_ID	X(10)	Cell ID of the B party receiving the call. Derivation: Optional.

Table 79-8 (Cont.) Associated GSM/Wireline Extension Record Fields

Name	Format	Description
A_TERM_CELL_ID	X(10)	Cell ID of the A party when the call terminated. Derivation: Optional.
CALL_REFERENCE	String	CallReference item. Optional

Supplementary Service Event Record (RECType 520)

This optional record is used for all non-call related supplementary service actions. The information attributable to a supplementary service event includes basic event information, location information, equipment information, and details of the supplementary service used.

The record applies only to mobile calls (GSM). Derived from the GSM item parameters as defined in TS GSM 12.05.

Table 79-9 describes the fields in the Supplementary Service Event Record.

Table 79-9 Supplementary Service Event Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 620 GSM/Wireline Extension Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.

Table 79-9 (Cont.) Supplementary Service Event Record Fields

Name	Format	Description
ACTION_CODE	H(1)	Qualifies the way in which the supplementary service is used. Values: 0: Registration 1: Erasure 2: Activation 3: Deactivation 4: Interrogation 5: Invocation 6: Registration of Password 9: Switch related Other values might apply according to the related original input format.

Table 79-9 (Cont.) Supplementary Service Event Record Fields

Name	Format	Description
SS_EVENT	H(2)	Uniquely defines the supplementary service or a group of supplementary services. Values: 00: All supplementary services 10: All line identification services 11: Calling number identification presentation 12: Calling number identification restriction 13: Connected number identification presentation 14: Connected number identification restriction 15: Malicious Call Identification 20: all call forwarding 21: Call forwarding unconditional 28: All conditional Call Forwarding 29: Call forwarding on mobile subscriber busy 2A: Call forwarding on no reply 2B: Call forwarding on subscriber not reachable 30: All call offering services 31: Call transfer 32: Mobile Access Hunting 40: all call completion services 41: Call waiting 42: Call hold 43: Completion of calls to busy subscribers 50: All multiparty services 51: multiparty service 60: All community of interest services 61: closed user groups 70: all charging supplement services 71: Advice of charge (charging) 72: Advice of charge (information) 80: All additional info transfer services 81: User to user signaling 90: All call barring 91: All Barring of outgoing Call Services 92: Barring of all outgoing calls 93: Barring of all outgoing international calls 94: Barring of all OG international except HPLMN 99: All Barring of incoming Call Services

Table 79-9 (Cont.) Supplementary Service Event Record Fields

Name	Format	Description
SS_EVENT (contd)	N/A	Uniquely defines the supplementary service or a group of supplementary services (contd). 9A: Barring of all incoming calls 9B: Barring of all IC calls when outside HPLMN all Switch related Services: 01...59: see related switch documentation (values used 1:1) Other values might apply according to the related original input format.
SS_PARAMETERS	String	Optional.
THIRD_PARTY_NUMBER	String	Optional.
CLIR_INDICATOR	Integer	Optional.
CHARGING_START_TIMESTAMP	Date	Optional.
CHARGING_END_TIMESTAMP	Date	Optional.
UTC_END_TIME_OFFSET	X(5)	Timezone where the call terminated. Derivation: Optional.
BASIC_SERVICE_CODE_LIST	String	Optional.

Associated Roaming Extension Record

Table 79-10 lists the fields in the Associated Roaming Extension Record.

Table 79-10 Associated Roaming Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	None
RECORD_NUMBER	Integer	None
TAP_FILE_SEQ_NO	Integer	None
RAP_FILE_SEQ_NO	Integer	None
RAP_RECORD_TYPE	String	None
SENDER	String	None
RECIPIENT	String	None
TAP_FILE_PATH	String	None
START_MISSING_SEQ_NUM	Integer	None
END_MISSING_SEQ_NUM	Integer	None
SUSPENSION_TIME	Date	None
PORT_NUMBER	String	None
TOTAL_TAX_REFUND	Decimal	None

Table 79-10 (Cont.) Associated Roaming Extension Record Fields

Name	Format	Description
TOTAL_DISCOUNT_REFUND	Decimal	None
GUARANTEED_BIT_RATE	String	None
MAXIMUM_BIT_RATE	String	None
HSCSD_INDICATO	String	None
SMS_ORIGINATOR	String	None
SMS_DESTINATION_NUMBER	String	None
DISCOUNTABLE_AMOUNT	Decimal	None
DISCOUNT_CODE	Integer	None
NETWORKACCESS_IDENTIFIE	String	None
ISM_SIGNALLING_CONTEXT	Integer	None
IMSI	String	None
HOME_BID	String	None
HOMELOCATION_DESCRIPTION	String	None
MOBILE_ID_NUMBER	String	None
MOBILE_DIR_NUMBER	String	None
TOTAL_ADVISEDCHARGE	Decimal	None
TOTAL_ADVISEDCHARGE_REFUND	Decimal	None
TOTAL_COMMISSION	Decimal	None
TOTAL_COMMISSION_REFUND	Decimal	None
ITEM_OFFSET	Integer	None
ERROR_CODE	Integer	None
TOTAL_SEVERE_RETURN_VALUE	Decimal	None
RETURN_DETAILS_COUNT	Integer	None
CLIR_INDICATOR	String	None

Associated RAP Extension Record

Table 79-11 lists the fields in the Associated RAP Extension Record.

Table 79-11 Associated RAP Extension Record Fields

Name	Format	Description
PATH_ITEMID	Integer	None
ITEM_OCCURRENCE	Integer	None
ITEM_LEVEL	Integer	None

Basic Service Event Record (RECType 520)

This optional record is used to store related TAP data.

The record applies only to mobile calls (GSM). Derived from the GSM item parameters as defined in TS GSM 12.05.

Table 79-12 lists the fields in the Basic Service Event Record.

Table 79-12 Basic Service Event Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 520 GSM/Wireline Extension Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
CHAIN_REFERENCE	String	Mandatory.
LONG_DURATION_INDICATOR	String	Mandatory.
BASIC_SERVICE	String	None
QOS_REQUESTED	String	None
QOS_USED	String	None
CHARGING_START_TIMESTAMP	Date	None
CHARGING_END_TIMESTAMP	Date	None
UTC_TIME_OFFSET	String	None
NUMBER_OF_UNITS	Decimal	None
WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	None
WHOLESALE_CHARGED_TAX_RATE	Integer	None
WHOLESALE_CHARGED_TAX_VALUE	Decimal	None
SPEECH_VERSION_REQUESTED	String	None
SPEECH_VERSION_USED	String	None
TRANSPARENCY_INDICATOR	String	None
FNUR	String	None
AIUR_REQUESTED	String	None
USER_PROTOCOL_INDICATOR	Integer	None
DATA_VOLUME_REFERENCE	String	None

Most-Called Information

This block contains the aggregated amount, duration, and occurrences of the most-called numbers. The number will be listed in the LIST attribute. The values listed in Table 79-13 can be used in EVAL expressions to give discounts based on most-called numbers.

Table 79-13 Most-Called Information Fields

Name	Format	Description
AMOUNT	Decimal	Aggregated amount.
COUNT	Decimal	Aggregated occurrences.
LIST	Integer	Number.
QUANTITY	String	Aggregated duration.

HSCSD Information Packet Record

This optional record is used to store related TAP data.

The record applies only to mobile calls (GSM). Derived from the GSM item parameters as defined in TS GSM 12.05.

High Speed Circuit Switched Data allows users subscribing to the General Bearer Service to use higher transmission rates by using multiple traffic channels simultaneously. This group element must contain Basic HSCSD parameters as at call setup and may also contain details of changes to those parameters.

Table 79-14 lists the fields in the HSCSD Information Packet Record.

Table 79-14 HSCSD Information Packet Record

Name	Format	Description
NUMBER_OF_CHANNELS	String	NumberOfChannels item. Mandatory.
CHANNEL_CODING_OK_LIST	Integer	ChannelCodingAcceptable list (comma-separated integers). Mandatory.
CHANNEL_CODING_USED	Integer	ChannelCoding item. Mandatory.
NUMBER_OF_CHANNELS_USED	Integer	NumberOfChannelsUsed item. Mandatory.
PM_LIST	Block	Optional. HSCSDParameterModification list.
AIUR	Integer	AiurRequested item.
MAX_NUMBER_OF_CHANNELS	Integer	NumberOfChannels item. Optional.
CHANNEL_CODING_USED	Integer	ChannelCoding item. Mandatory.

Table 79-14 (Cont.) HSCSD Information Packet Record

Name	Format	Description
NUMBER_OF_CHANNELS_USED	Integer	NumberOfChannelsUsed item. Mandatory.
INITIATING_PARTY	Integer	InitiatingParty item. Mandatory.
MODIFICATION_TIMESTAMP_	Date	ModificationTimestamp item. Mandatory.
UTC_TIME_OFFSET	String	NumberOfChannels item.

Associated GPRS Extension Record (RECType 540)

This record stores GPRS service information. This record is optional and will be generated only if the related Basic Detail Record indicates a GPRS service.

Table 79-15 describes the fields in the Associated GPRS Extension Record.

Table 79-15 Associated GPRS Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 540 Associated GPRS Extension Record
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PORT_NUMBER	X(24)	Identifies the customer IMSI or SIM number. Option: For Value Added Services and APLMN Service Center Usage, either the IMSI or MSISDN might be supplied, although one of them must be supplied and, where available, the IMSI is preferred. For normal mobile calls, the SIM number is preferred. Derivation: Optional. From the GSM item served IMSI as defined in TS GSM 12.05. Defined in TS GSM 03.03.

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
DEVICE_NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Condition: Not present where the terminal equipment is not involved in the call; for example, in forwarded call cases. It is not mandatory for the VPLMN to transfer this information. Derivation: Optional. From the GSM item IMEI as defined in TS GSM 12.05. Defined in TS GSM 03.03. **Note:** Even though the IMEI is 16 digits in length, the check digit is not transmitted.
A_NUMBER_USER	X(40)	The customer who owns the number from which the call was originated, for terminated calls. Not used for rating, but could be used on invoices. Condition: There is no calling number present where it is unavailable. Could be different from the A Number; for example, in case of VPN calls. For VPN calls, the A Number contains the party to be billed, and this field contains the user initiating the call. Values: See A_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.
DIALED_DIGITS	X(40)	The number dialed by the customer when establishing a call or the number to which the call is forwarded or transferred. Can be used for managing disputes. Condition: There might be no called number for the basic service emergency call but operators might optionally insert the digits 112 or their national emergency number into this field. The notation should always be local; for example, 04106768124. Values: See B_NUMBER. Derivation: Optional. From the GSM item Calling Number as defined in TS GSM 12.05. This item is of type Address String and is further expanded into the items type of number, numbering plan and the number sent across the air-interface as defined in TS GSM 04.08 and 09.02 or in international notation. Set by the first processor and left unchanged.

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
VAS/PRODUCT_CODE	X(10)	A classification of Value Added Services as generated by the sender. Can be used to map to a specific internal service code to implement specific usage scenarios for any rating purposes. Values: VMAIL: Voice Mail Services SEC: Secretarial Services OPER: Telephonic Operator Services FI: Financial Information TRAVEL: Travel Information This is not a definitive list and might be added to through MoU-TADIG from time to time or might be user defined. Derivation: Optional. From the GSM item vasCode as defined in GSM TD17. Set by the first processor and left unchanged.
ORIGINATING_SWITCH_IDENTIFICATION	X(15)	Identifies the MSC or SwitchID handling the origin of the call. In case of mobile roaming calls (GSM), this field contains the MOC-related MCC/MNC. In case of wireline networks, this field contains the primary switch that generated this CDR. Can be used by any interconnect rating processor to uniquely identify the trunk names but will only be used if the trunk names are only unique within the related switch. See TRUNK_INPUT and TRUNK_OUTPUT. Can also be used to normalize the A Number for MOC roaming. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: • The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. • The signaling point code as per GSM 03.03; for example, 253464. • The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201. • MCC = Mobile Country Code; MNC = Mobile Network Code • A name; for example, "HELSINKI": Must be uppercase. • A switchID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case of interconnect records (for intercarrier rating/billing reasons,) or for mobile (roaming) records. Set by the first processor and left unchanged. The switch might not be needed if the trunk names are unique within the total network).

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
TERMINATING_SWITCH_IDENTIFICATION	X(15)	Identifies the MSC or Switch ID handling the termination of the call. In case of mobile roaming calls (GSM), this field contains the MTC-related MCC/MNC. In case of wireline networks, this field contains the secondary switch or is empty. Can be used to normalize the B Number in case of MTC-roaming cause. In case of roaming, this field contains the MCC/MNC. Encoding: Encoded as one of the following according to the requirements of the sender: • The MSISDN of the MSC as per GSM 03.03; for example, 44836100456. • The signaling point code as per GSM 03.03; for example, 253464. • The MCC/MNC (TADIG, PLMN) for mobile roaming calls; for example, 26201. • MCC = Mobile Country Code; MNC = Mobile Network Code • A name; for example, "HELSINKI": Must be uppercase. • A switch ID as set up within a local fixed network structure. Derivation: Optional, only mandatory in case of mobile (roaming) records. Set by the first processor and left unchanged.
MS_CLASS_MARK	9(1)	The power capability of the equipment making or receiving the call. Mobiles and transmobiles usually have class 2 capability, handhelds class 4, and PCN applications class 5. Some transmobiles have reduced capability and are classified as class 3. Usually not used. Condition: Only available if supported by the network and the call originates or terminates from the equipment. Is not available in call forwarding cases. Values: 1. Class Mark 2 2. Class Mark 3 3. Class Mark 4 4. Class Mark 5 Other values might apply according to the related original input format. Derivation: Optional. From the GSM item msclassmark as defined in TS GSM 12.05. Set by the first processor and left unchanged.
ROUTING_AREA	X(10)	Routing Area at the time of record creation (S-CDR only).

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
LOCATION_AREA_INDICATOR	X(10)	Identifies the MSC responsible for handling the call and the location of the equipment making or receiving the call. The definition of these items can be found in the Data Dictionary under MSC Identification, Location Area, and Cell Id. Can be used to map to a specific internal service code to implement a event-dependent rating. Condition: Is not available if not supported by the network or the call does not terminate at the equipment; for example, in call forwarding cases. Values: The Location Area Code is a two-octet string as defined in TS GSM 04.08. For the TAP, the octets are converted to a decimal number in the range 0x00000000 to 0xFFFFFFFF. Derivation: Optional. From the GSM item locationAreaCode as defined in TS GSM 12.05 or directly taken from the sender (VAS). Set by the first processor and left unchanged.
CHARGING_ID	Decimal	PDP context identifier used to identify this PDP context in different records created by GSNs. This field is a charging identifier which can be used together with GGSN address to identify all records produced in SGSN(s) and GGSN involved in a single PDP context. Charging ID is generated by GGSN at PDP context activation and transferred to context requesting SGSN. At inter-SGSN routing area update, charging ID is transferred to the new SGSN as part of each active PDP context. Different GGSNs allocate the charging ID independently of each other and might allocate the same numbers. The CGF and/or BS might check the uniqueness of each charging ID together with the GGSN address and optionally (if still unambiguous) with the record opening timestamp.
SGSN_ADDRESS	X(64)	Current SGSN Address used. Optional.
GGSN_ADDRESS	X(64)	IP Address of the GGSN currently used. Optional.
WLAN_ADDRESS	String	Optional.
APN_ADDRESS	X(64)	The logical name of the connected access point to the external packet data network. APN comprises network identifier and operator identifier. This field contains the logical Access Point Name used to determine the actual connected access point. APN comprises network identifier and operator identifier. APN can also be a wildcard, in which case SGSN selects the access point address. See GSM 03.03 [4] and GSM 03.60 [8] for more information about APN format and access point decision rules.
NODE_ID	X(64)	Name of the recording entity; for example, could be the charging gateway name.
TRANS_ID	9(10)	Sequence number which the recording entity generates (NODE_ID). The number is allocated sequentially including all CDR types. It links together the CDR of a same recording entity.
SUB_TRANS_ID	9(10)	Partial record sequence number. This field contains a running sequence number which links the partial records generated for a PDP context/GPRS session. It can be used in post-processing to detect missing CDRs for a GPRS session. It links together the CDRs/events of a same session.

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
NETWORK_INITIATED_PDP	9(1)	Network Initiated PDP context. The network initiates a context when it calls an ME. Values: 0: False 1: True
PDP_TYPE	X(4)	Defines the PDP type; for example, X.25, IP, PPP, or IHOSS:OSP (see GSM 09.60 for exact format).
PDP_ADDRESS	X(64)	PDP address of the served IMSI (Ipv4, IPv6, X.121).
PDP_REMOTE_ADDRESS	X(255)	List of PDP address of remote host (comma-separated value, G-CDR only, X25 only).
PDP_DYNAMIC_ADDRESS	9(1)	Indicates that the PDP address has been dynamically allocated for that particular PDP context. This field is missing if address is static; for example, part of PDP context subscription. Dynamic address allocation might be relevant for charging; for example, the duration of PDP context as one balance element offered and possibly owned by network operator. Values: 0: False 1: True
DIAGNOSTICS	X(255)	Includes a more detailed technical reason for the release of the connection and might contain one of the following: • A MAP error from GSM 09.02 [17] • A Cause from GSM 04.08 [16] The diagnostics might also be extended to include manufacturer and network-specific information.
CELL_ID	X(10)	The cell from which the call originated. Can be used to identify the location of the caller. Condition: Operators might not transfer the cell identity. Only available if the call originates or terminates from a mobile phone; for example, not available in call divert cases. Values: The cell identity is a two-octet string as defined in TS GSM 04.08. However, an original hex value is copied. Derivation: Optional. From the GSM item Cell Id as defined in TS GSM 12.05. Set by the first processor and left unchanged.
CHANGE_CONDITION	9(1)	The condition that triggers the creation of this volume container as defined by ETSI. Values: 0: Quality of Service Change 1: Tariff Change 2: Record Closed

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
QoS_REQUESTED_PRECEDENCE	X(1)	The priority applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Unspecified 1: High Priority 2: Normal Priority 3: Low Priority Derivation: GSM item QoS Precedence (GSM 12.15).
QoS_REQUESTED_DELAY	X(1)	The transfer delay applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Delay class 1 1: Delay class 2 2: Delay class 3 3: Delay class 4 Derivation: GSM item QoSDelay (GSM 12.15).
QoS_REQUESTED_RELIABILTY	X(1)	The reliability applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Unspecified Reliability 1: Acknowledged GTP 2: Unacknowledged GTP/acknowledged LLC 3: Unacknowledged GTP/ acknowledged RLC 4: Unacknowledged GTP/LLC/RLC 5: Unacknowledged unprotected data Derivation: GSM item QoS Reliability (GSM 12.15).

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
QoS_REQUESTED_PEAK_THROUGHPUT	X(2)	The peak throughput applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Unspecified 1: Up to 100 octets per seconds 2: Up to 200 octets per seconds 3: Up to 400 octets per seconds 4: Up to 800 octets per seconds 5: Up to 1600 octets per seconds 6: Up to 3200 octets per seconds 7: Up to 6400 octets per seconds 8: Up to 12800 octets per seconds 9: Up to 25600 octets per seconds Derivation: GSM item QoS Peak Throughput (GSM 12.15).
QoS_REQUESTED_MEAN_THROUGHPUT	X(2)	The mean throughput applicable to a GPRS connection. Condition: Mandatory within groups GSM Quality Of Service Requested. Values: 0: Best Effort 1: Mean 100 octets per hour 2: Mean 200 octets per hour 3: Mean 500 octets per hour 4: Mean 1000 octets per hour 5: Mean 2000 octets per hour 6: Mean 5000 octets per hour 7: Mean 10000 octets per hour 8: Mean 20000 octets per hour 9: Mean 50000 octets per hour 10: Mean 100000 octets per hour 11: Mean 200000 octets per hour 12: Mean 500000 octets per hour 13: Mean 1000000 octets per hour 14: Mean 2000000 octets per hour 15: Mean 5000000 octets per hour 16: Mean 10000000 octets per hour 17: Mean 20000000 octets per hour 18: Mean 50000000 octets per hour Derivation: GSM item QoS Mean Throughput (GSM 12.15).
QoS_USED_PRECEDENCE	X(1)	Quality of Service Precedence class. See QoS_REQUESTED_PRECEDENCE.

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
QoS_USED_DELAY	X(1)	QOS delay class, defined by ETSI. See QoS_REQUESTED_DELAY.
QoS_USED_RELIABILTY	X(1)	QOS reliability class, defined by ETSI. See QoS_REQUESTED_RELIABILTY.
QoS_USED_PEAK_THROUGHPUT	X(2)	QOS peak throughput class, defined by ETSI. See QoS_REQUESTED_PEAK_THROUGHPUT.
QoS_USED_MEAN_THROUGHPUT	X(2)	QOS mean throughput class, defined by ETSI. See QoS_REQUESTED_MEAN_THROUGHPUT.
NETWORK_CAPABILITY	X(10)	MS network capability information element of the served MS on PDP context activation or on GPRS attachment as defined in GSM 04.08 [16]. Condition: Optional. Derivation: GSM item network capability (GSM 04.08 [16]).
SGSN_CHANGE	9(1)	Indicates that this is the first record after an inter-SGSN routing area update. Condition: Mandatory. Values: 0: default, if this is not the 1st record 1: indicates the first record after an inter SGSN-change
START_SEQUENCE_NO	String	Optional.
END_SEQUENCE_NO	String	Optional.
B_CELL_ID	X(10)	Cell ID of the B party receiving the call. Derivation: Optional.
A_TERM_CELL_ID	X(10)	Cell ID of the A party when the call terminated. Derivation: Optional.
PDP_CONTEXT_START_TIMESTAMP	Date	Conditional in TAP 3.10. Indicates the start time of the PDP context when the Call Event Details (GPRS Call) represents an intermediate or last partial of a PDP context. Used in TAP files. Format: *CCYYMMDDHHMMSS*
PDP_UTC_TIME_OFFSET	String	Conditional in TAP 3.10. Indicates the UTC time offset for PDP_CONTEXT_START_TIMESTAMP.

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
SERVICE_USED_CHARGING_START_TIMESTAMP	Date	Conditional in TAP 3.10. Indicates the start time for charging GPRS calls. This field is present when the value is not the same as the associated Call Event Start Timestamp field (DETAIL.ASS_GPRS_EXT.GS_PACKET.CHARGING_START_TIMESTAMP). Used in TAP files. Format: *CCYYMMDDHHMMSS*
SERVICE_USED_UTC_TIME_OFFSET	String	Conditional in TAP 3.10. Indicates the UTC time offset for SERVICE_USED_CHARGING_START_TIMESTAMP.
TYPE_OF_CONTROLLING_NODE	Integer	Conditional in TAP 3.10.
GPRS_SERVICE_USAGE_PACKET	Block	*n* times. Optional. Mandatory.
CHARGING_START_TIMESTAMP	Date	Optional.
CHARGING_END_TIMESTAMP	Date	Optional.
UTC_TIME_OFFSET	String	Optional.
QOS_REQUESTED_PRECEDENCE	String	Optional.
QOS_REQUESTED_DELAY	String	Optional.
QOS_REQUESTED_RELIABILITY	String	Optional.
QOS_REQUESTED_PEAK_THROUGHPUT	String	Optional.
QOS_REQUESTED_MEAN_TRHOUGHPUT	String	Optional.
QOS_USED_PRECEDENCE	String	Optional.
QOS_USED_DELAY	String	Optional.
QOS_USED_RELIABILITY	String	Optional.
QOS_USED_PEAK_TRHOUGHPUT	String	Optional.
QOS_USED_MEAN_TRHOUGHPUT	String	Optional.
VOLUME_RECEIVED	Decimal	Mandatory.
VOLUME_SENT	Decimal	Mandatory.

Table 79-15 (Cont.) Associated GPRS Extension Record Fields

Name	Format	Description
UMTS_QOS_REQUESTED	String	Optional. Identifies the UMTS Quality of Service requested for GPRS calls. Used in TAP files. Format: This field contains comma-separated tag-value pairs of the following TAP fields with their respective tags as shown below: • QoS Traffic Class: 1 • QoS Max Bit Rate Uplink: 2 • Qos Max Bit Rate Downlink: 3 • Qos Guaranteed Bit Rate Downlink: 4 • Qos Guaranteed Bit Rate Uplink: 5 • Qos Allocation Retention Priority: 6 The fields QoS Traffic Class, QoS Max Bit Rate Uplink, Qos Max Bit Rate Downlink are Mandatory. The others are optional. Example 1: If the TAP field values are as follows: • QoS Traffic Class: **3** • QoS Max Bit Rate Uplink: **63** • Qos Max Bit Rate Downlink: **128** • Qos Guaranteed Bit Rate Downlink: **61** • Qos Guaranteed Bit Rate Uplink: **250** • Qos Allocation Retention Priority: **3** The value of the EDR field is **1,3,2,63,3,128,4,61,5,250,6,3** Example 2: If the TAP field values are as follows: QoS Traffic Class: **2** QoS Max Bit Rate Uplink: **56** Qos Max Bit Rate Downlink: **128** Qos Guaranteed Bit Rate Uplink: **250** The value of the EDR field is: **1,2,2,56,3,128,5,250**
UMTS_QOS_USED	String	Optional. Identifies the UMTS Quality of Service used for GPRS calls. Used in TAP files. The description for this field is identical to the description for UMTS_QOS_REQUESTED.

Associated WAP Extension Record (RECType 570)

Stores information for WAP events. This record is optional and will only be generated if the related Basic Detail Record indicates a WAP service.

Table 79-16 describes the fields in the Associated WAP Extension Record.

Table 79-16 Associated WAP Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 570 Associated WAP Extension Record
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PORT_NUMBER	X(24)	Identifies the customer IMSI or SIM number. Option: For Value Added Services and APLMN Service Center Usage either the IMSI or MSISDN might be supplied, although one of them must be supplied and, where available, the IMSI is preferred. For normal mobile calls, the SIM number is preferred. Derivation: Optional. From the GSM item served IMSI as defined in TS GSM 12.05. Defined in TS GSM 03.03.
DEVICE_NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Condition: Not present where the terminal equipment is not involved in the call; for example, in forwarded call cases. It is not mandatory for the VPLMN to transfer this information. Derivation: Optional. From the GSM item IMEI as defined in TS GSM 12.05. Defined in TS GSM 03.03. **Note:** Even though the IMEI is 16 digits in length, the check digit is not transmitted.
SESSION_ID	X(64)	Session ID as provided by the WAP gateway.
RECORDING_ENTITY	X(64)	Name of the recording Entity; for example, the WAP gateway or mediation device.
TERMINAL_CLIENT_ID	X(64)	The served WAP terminal client ID (WAP gateway user identity).
TERMINAL_IP_ADDRESS	X(64)	IP address of the WAP terminal.
DOMAIN_URL	X(255)	URL implementing the service.
BEARER_SERVICE	X(3)	See BASIC_SERVICE.
BEARER_SERVICE_CODE	X(2)	See SERVICE_CODE.

Table 79-16 (Cont.) Associated WAP Extension Record Fields

Name	Format	Description
HTTP_STATUS	9(3)	HTTP status code from the origin server or servlet. Values: 100: CONTINUE 101: SWITCHING_PROTOCOLS 200: SUCCESS 201: CREATE 202: ACCEPTED 203: NON-AUTHORITATIVE_INFORMATION 204: NO_CONTENT 205: RESET_CONTENT 206: PARTIAL_CONTENT 300: MULTIPLE_CHOICE 301: MOVED_PERMANENTLY 302: FOUND 303: SEE_OTHER 304: NOT_MODIFIED 305: USE_PROXY 307: TEMPORARY_REDIRECT 400: BAD_REQUEST 401: UNAUTHORIZED 402: PAYMENT_REQUIRED 403: FORBIDDEN 404: NOT_FOUND 405: METHOD_NOT_ALLOWED 406: NOT_ACCEPTABLE 407: PROXY_AUTHENTICATION_REQUIRED 408: REQUEST_TIMEOUT 409: CONFLICT 410: GONE 411: LENGTH_REQUIRED 412: PRECONDITION_FAILED 413: REQUEST_ENTITY_TOO_LARGE 414: REQUEST-URI_TOO_LONG 415: UNSUPPORTED_MEDIA_TYPE 416: REQUESTED_RANGE_NOT_SATISFIABLE 417: EXPECTATION_FAILED 500: INTERNAL_SERVER_ERROR 501: NOT_IMPLEMENTED 502: BAD_GATEWAY 503: SERVICE_UNAVAILABLE 504: GATEWAY_TIMEOUT 505: HTTP_VERSION_NOT_SUPPORTED

Table 79-16 (Cont.) Associated WAP Extension Record Fields

Name	Format	Description
WAP_STATUS	9(3)	The WSP/WAP status code. Values: 16: CONTINUE 17: SWITCHING_PROTOCOLS 20: OK, SUCCESS 33: CREATED 34: ACCEPTED 35: NON-AUTHORITATIVE_INFORMATION 36: NO_CONTENT 37: RESET_CONTENT 38: PARTIAL_CONTENT 48: MULTIPLE_CHOICE 49: MOVED_PERMANENTLY 50: MOVED_TEMPORARILY 51: SEE_OTHER 52: NOT_MODIFIED 53: USE_PROXY 55: TEMPORARY_REDIRECT 64: BAD_REQUEST 65: UNAUTHORIZED 66: PAYMENT_REQUIRED 67: FORBIDDEN 68: NOT_FOUND 69: METHOD_NOT_ALLOWED 70: NOT_ACCEPTABLE 71: PROXY_AUTHENTICATION_REQUIRED 72: REQUEST_TIMEOUT 73: CONFLICT 74: GONE 75: LENGTH_REQUIRED 76: PRECONDITION_FAILED 77: REQUEST_ENTITY_TOO_LARGE 78: REQUEST-URI_TOO_LONG 79: UNSUPPORTED_MEDIA_TYPE 80: REQUESTTED_RANGE_NOT_SATISFIABLE 81: EXPECTATION_FAILED 96: INTERNAL_SERVER_ERROR 97: NOT_IMPLEMENTED 98: BAD_GATEWAY 99: SERVICE_UNAVAILABLE 100: GATEWAY_TIMEOUT 101: HTTP_VERSION_NOT_SUPPORTED

Table 79-16 (Cont.) Associated WAP Extension Record Fields

Name	Format	Description
ACKNOWLEDGE_STATUS	9(1)	Acknowledge status of the response. Values: 1: OK acknowledgment has been received. 2: Response terminated by the server. 3: Response terminated by the terminal. 4: Acknowledgment has not been received. 5: Acknowledgment is not used with this connection type.
ACKNOWLEDGE_TIME	YYYYMMDD HHMISS	Time of the acknowledgment.
EVENT_NUMBER	X(60)	Assigned user event number as generated by the WAP gateway.
GGSN_ADDRESS	X(64)	IP Address of the GGSN currently used.
SERVER_TYPE	X(64)	A description of the type of server providing the service.
CHARGING_ID	Decimal	PDP context identifier used to identify this PDP context in different records created by GSNs. This field is a charging identifier which can be used together with GGSN address to identify all records produced in SGSN(s) and GGSN involved in a single PDP context. Charging ID is generated by GGSN at PDP context activation and transferred to context requesting SGSN. At inter-SGSN routing area update, charging ID is transferred to the new SGSN as part of each active PDP context. Different GGSNs allocate the charging ID independently of each other and might allocate the same numbers. The CGF and/or BS might check the uniqueness of each charging ID together with the GGSN address and optionally (if still unambiguous) with the record opening timestamp.
WAP_LOGIN	X(24)	Login used during the WAP session. This might occur in addition to the MSISDN; for example, this field might contain a user name of a session which has been opened within a WAP session. Condition: Optional. Might be mandatory for specific WAP scenarios. Derivation: Set by the first processor and left unchanged.
IDENTIFIER	String	Mandatory.
TYPE	Integer	Mandatory.

Associated CAMEL Extension Record (RECType 700)

In the following associated record of the sol42 format extended CAMEL service information could be stored. This record is optional and is attached to any other associated service extension record.

Table 79-17 lists the Associated CAMEL Extension Record fields.

Table 79-17 Associated CAMEL Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long Value: 700: Associated CAMEL Extension Record
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
SERVER_TYPE_OF_NUMBER	Z(1)	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
SERVER_NUMBERING_PLAN	X(1)	Optional.
SERVER_ADDRESS	X(40)	Identifies the server interrogated. Mandatory.
SERVICE_LEVEL	Z(1)	Identifies the level of CAMEL service provided [0-3]. Mandatory.
SERVICE_KEY	Z(10)	Identifies the CAMEL service Logic to be applied. Mandatory.
DEFAULT_CALL_HANDLING_INDICATOR	Z(1)	Indicates whether or not a CAMEL call encountered default handling. Values: 0: Continue the call 1: Release the call
MSC_TYPE_OF_NUMBER	Z(1)	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
MSC_NUMBERING_PLAN	X(1)	Optional.
MSC_ADDRESS	X(40)	Identifies the MSC that generated the CAMEL reference number (might be different from SERVER_ADDRESS). Mandatory.
CAMEL_REFERENCE_NUMBER	X(20)	In association with the MSC_ADDRESS, provides a unique identifier for each CAMEL invocation. Mandatory.
CAMEL_INITIATED_CF_INDICATOR	Z(1)	Optional, but should be defaulted to 0 (1=CAMEL call forwarding).
CAMEL_MODIFICATION_LIST	X(20)	Optional, comma-separated string of integers.
DEST_GSMW_TYPE_OF_NUMBER	Z(1)	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
DEST_GSMW_NUMBERING_PLAN	X(1)	Optional.

Table 79-17 (Cont.) Associated CAMEL Extension Record Fields

Name	Format	Description
DEST_GSMW_NUMBER	X(40)	Optional, used to identify CAMEL redirection destination (could contain an MSISDN, IP, LOGIN, etc.) when the primary extension is of type GSM (for example, ASS_GSMW_EXT).
DEST_GSMW_NUMBER_ORIGINAL	X(40)	Optional, DEST_GSM_NUMBER as received (before normalization).
DEST_GPRS_APN_ADDRESS	X(64)	Optional (but might be mandatory for specific zoning scenarios; for example, if an apn_group is used) when the primary extension is of type GPRS (ie., ASS_GPRS_EXT).
DEST_GPRS_PDP_REMOTE_ADDRESS	X(255)	Optional.
CSE_INFORMATION	X(40)	Optional, the information downloaded by the CAMEL server.
GSM_CALL_REFERENCE_NUMBER	X(20)	Optional.
EXCHANGE_RATE	Decimal	Contains the exchange rate which has been used to convert the Incoming currency to the internal currency as indicated in the field CHARGED_CURRENCY_TYPE. Can be used to convert the virtual currency SDR (which is used in conjunction of TAP) to internal currencies and convert the Charge back to SDR after Rating. This would be a typical usage for Interconnection Rating. Values: Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Optional, defaulted 00000000001 (=1,00).

Associated Suspense Extension Record (RECType 720)

Table 79-18 describes the fields in the Associated Suspense Extension Record. This record is optional and can appear once.

Table 79-18 Associated Suspense Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = **720**
RECORD_NUMBER	Integer	Mandatory. Auto-generated.

Table 79-18 (Cont.) Associated Suspense Extension Record Fields

Name	Format	Description
SUSPENSE_STATUS	Integer	Mandatory. Calculated.
SUSPENSE_REASON	Integer	The suspense reason. Mapped from the error code. Mandatory. Calculated.
SUSPENSE_SUBREASON	Integer	Mandatory. Calculated.
RECYCLE_KEY	String	Search key for choosing EDRs to recycle. Optional.
ERROR_CODE	Integer	Mandatory. Calculated.
SUSPENSE_ID	Integer	Original suspense POID ID. Mandatory when recycling.
PIPELINE_NAME	String	The name of the pipeline, derived from the pipeline registry. Mandatory when recycling. Calculated.
SOURCE_FILENAME	String	The source file name. The same as INTERNAL.STREAM_NAME. Mandatory. Calculated.
SERVICE_CODE	String	Equal to DETAIL.INTERN_SERVICE_CODE. Mandatory. Calculated.
EDR_RECORD_TYPE	String	Equal to DETAIL.RECORD_TYPE. Mandatory. Calculated.
EDR_BUF	String	A stored representation of the EDR container including fields overwritten and enriched by the pipeline. Mandatory when recycling. Calculated.
UTC_OFFSET_SECONDS	Integer	This value enables Suspense Management Center to represent call record times using the same time zone used in the records themselves. The value represents the offset between the time zone of the EDR and UTC in seconds. Mandatory. Calculated.
EDR_SIZE	Integer	The size of DETAIL.ASS_SUSPENSE_EXT.EDR_BUF. Mandatory. Calculated.
QUERYABLE_FIELDS	String	The queryable field values defined in the registry. Separated by tab characters. Mandatory. Calculated.
OVERRIDE_REASONS	String	Optional. May be set equal to the override reason code during recycling.
ACCOUNT_POID	String	Optional. Calculated.
SUSPENDED_FROM_BATCH_ID	String	Optional. Calculated.
PIPELINE_CATEGORY	String	Mandatory. Calculated.
RECYCLING_MODE	Integer	Mandatory. Calculated. Equal to DETAIL.INTERN_PROCESS_STATUS

Associated Content Extension Record (RECType 550)

This optional record is used to store related TAP data. Table 79-19 describes the fields in the Associated Content Extension Record.

Table 79-19 Associated Content Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 550.
RECORD_NUMBER	Integer	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
FRAUD_MONITOR_INDICATOR	String	Optional, but should be defaulted to '0'. Could be used by some modules to perform number-normalization.
RAP_FILE_SEQ_NO	String	Optional.
ORDER_PLACED_TIMESTAMP	Date	Optional.
ORDER_PLACED_UTC_TIME_OFFSET	String	Mandatory if ORDER_PLACED_TIMESTAMP is present.
REQUESTED_DELIVERY_TIMESTAMP	Date	Optional.
REQ_DELIVERY_UTC_TIME_OFFSET	String	Mandatory if REQUESTED_DELIVERY_TIMESTAMP is present.
ACTUAL_DELIVERY_TIMESTAMP	Date	Optional.
ACT_DELIVERY_UTC_TIME_OFFSET	String	Mandatory if ACTUAL_DELIVERY_TIMESTAMP is present.
TOTAL_TRANSACTION_DURATION	Integer	Optional.
TRANSACTION_STATUS	Integer	Optional.
CHARGED_PARTY_INFO	Block	Charged party information block. Mandatory.
ID_LIST	N.A.	ChargedPartyId list. Mandatory.
HOMEID_LIST	N.A.	ChargedPartyHomeId list. Optional.
LOCATION_LIST	N.A.	ChargedPartyLocation list. Optional.
EQUIPMENT	N.A.	ChargedPartyEquipment block. Optional.
SERVING_PARTIES_INFO	Block	ServingPartiesInformation block. Mandatory.
PROVIDER_LIST	N.A.	ContentProviderId list. Optional.

Table 79-19 (Cont.) Associated Content Extension Record Fields

Name	Format	Description
ISP_LIST	N.A.	InternetServiceProviderId list. Optional.
NETWORK_LIST	N.A.	Network list. Optional.
SERVICE_USED_LIST	N.A.	ContentServiceUsed list. Mandatory.
CONTENT_TRANSACTION_CODE	Integer	ContentTransactionCode item. Mandatory.
OBJECT_TYPE	Integer	ObjectType item. Optional.
TRANSACTION_DESCRIPTION_SUPP	Integer	TransactionDescriptionSupp item. Optional.
TRANSACTION_DETAIL_DESCRIPTION	String	TransactionDetailDescription item. Optional.
TRANSACTION_IDENTIFIER	String	TransactionIdentifier item. Mandatory.
TRANSACTION_AUTH_CODE	String	TransactionAuthCode item. Optional.
DATA_VOLUME_INCOMING	Integer	DataVolumeIncoming item. Optional.
DATA_VOLUME_OUTGOING	Integer	DataVolumeOutgoing item. Optional.
TOTAL_DATA_VOLUME	Integer	TotalDataVolume item. Optional.
CHARGE_REFUND_INDICATOR	Integer	ChargeRefundIndicator item. Optional.
CONTENT_CHARGING_POINT	Integer	ContentChargingPoint item. Optional.
PAID_INDICATOR	Integer	PaidIndicator item. Optional.
PAYMENT_METHOD	Integer	PaymentMethod item. Optional.
ADVISED_CHARGE_CURRENCY	String	AdvisedChargeCurrency item. Optional.
ADVISED_CHARGE	Decimal	AdvisedCharge item. Optional. Mandatory in the AdvisedChargeInformation block.
COMMISSION	Decimal	Commission item. Optional.

Associated Location Extension Record

The OutGrammar stores information of Content and Location from the EDR container into the output TAP blocks. This is performed using ASN calls of iScript in TAP version 3.10 OutGrammar.

Table 79-20 describes the fields in the Associated Location Extension Record.

Table 79-20 Associated Location Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Must be set to 560.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
FRAUD_MONITOR_INDICATOR	String	FraudMonitorIndicator item. Optional.
RAP_FILE_SEQ_NO	String	RapFileSequenceNumber item. Optional.
REC_ENTITY_CODE	Integer	RecEntityCode item. Mandatory.
CALL_REFERENCE	String	CallReference item. Optional.
GMLC_ADDRESS	String	N/A
TRACKING_CUSTOMER_INFORMATION	Block	TrackingCustomerInformation block. Optional.
ID_LIST	N.A.	TrackingCustomerId list. Mandatory.
HOME_ID_LIST	N.A.	TrackingCustomerHomeId list. Mandatory.
LOCATION_LIST	N.A.	TrackingCustomerLocation list. Mandatory.
EQUIPMENT	N.A.	TrackingCustomerEquipment block. Optional.
LCS_SP_INFORMATION LCSSP_INFO	Block	LCSSPInformation block. Optional.
ID_LIST	N.A.	LCSSPId list. Mandatory.
ISP_LIST	N.A.	InternetServiceProviderId list. Optional.
NETWORK_LIST	N.A.	Network list. Optional.
TRACKED_CUSTOMER_INFORMATION	Block	TrackedCustomerInformation block. Optional.
ID_LIST	N.A.	TrackedCustomerId list. Mandatory.

Table 79-20 (Cont.) Associated Location Extension Record Fields

Name	Format	Description
HOME_ID_LIST	N.A.	TrackedCustomerHomeId list. Mandatory.
LOCATION_LIST	N.A.	TrackedCustomerLocation list Mandatory.
EQUIPMENT	N.A.	TrackedCustomerEquipment block. Optional.
LOCATION_SERVICE_USAGE	Block	LocationServiceUsage block. Mandatory.
LCSQosRequested	Block	Mandatory.
LCS_REQUEST_TIMESTAMP	Date	Mandatory.
LCS_REQ_UTC_OFFSET	String	LCSRequestTimestamp item. Mandatory.
H_ACCURACY_REQUESTED	Integer	HorizontalAccuracyRequested item. Optional.
V_ACCURACY_REQUESTED	Integer	VerticalAccuracyRequested item. Optional.
RESPONSE_TIME_CATEGORY	Integer	ResponseTimeCategory item. Optional.
TRACKING_PERIOD	Integer	TrackingPeriod item. Optional.
REQ_TRACKING_FREQUENCY	Integer	TrackingFrequency (requested) item. Optional.
LCSQosDelivered	Block	Optional.
LCS_TRANS_STATUS	Integer	LCSTransactionStatus item. Optional.
H_ACCURACY_DELIVERED	Integer	HorizontalAccuracyDelivered item. Optional.
V_ACCURACY_DELIVERED	Integer	VerticalAccuracyDelivered item. Optional.
RESPONSE_TIME	Integer	ResponseTime item. Optional.
POSITIONING_METHOD	Integer	PositioningMethod item. Optional.
DEL_TRACKING_PERIOD	Integer	TrackingPeriod item. Optional.
DEL_TRACKING_FREQUENCY	Integer	TrackingFrequency (delivered) item. Optional.
AGE_OF_LOCATION	Integer	AgeOfLocation item. Optional.

Table 79-20 (Cont.) Associated Location Extension Record Fields

Name	Format	Description
CHARGING_TIMESTAMP	Date	ChargingTimeStamp item. Optional.
CHARGING_UTC_OFFSET	String	ChargeInformationList data is stored in DETAIL.ASSOCIATED_CHARGE_BREAKDOWN.CHARGE_PACKET. Mandatory if LCSRequestTimestamp is given.

Associated Value Added Service (VAS) Extension Record (RECType 710)

A Value Added Service (VAS) item represents usage of value added services outside of a standard call; i.e., unrelated to either a Mobile Originated Call or a Mobile Terminated Call. VAS consists of Chargeable Subscriber and Value Added Service Used, which are mandatory; conditionally, Network Type and RAP File Sequence Number; optionally, Operator Specific Information.

Table 79-21 describes the fields in the Associated Value Added Service Extension Record.

Table 79-21 Associated Value Added Service (VAS) Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 710.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
VAS_CODE	Integer	Mandatory.
VAS_SHORT_DESC	String	Optional.
VAS_DESC	String	Optional.
CHARGING_START_TIMESTAMP	Date	Optional.
CHARGING_END_TIMESTAMP	Date	Optional.
UTC_TIME_OFFSET	String	Optional.

Associated BRM Balance Record (RECType 900)

Stores data to be loaded into the BRM database.

Associated BRM Billing Records might occur more than once for each Basic Detail Record. This is the case if more than one balance is affected by one event.

Table 79-22 describes the fields in the Associated BRM Balance Record.

Table 79-22 Associated BRM Balance Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Values: 900: Associated BRM Balance Record Usage: Determination of the different record types. Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
ACCOUNT_POID	X(255)	POID of the account. Example: 1 /account 123456789 0 Derivation: Mandatory.
SERVICE_POID	X(255)	POID for the service. Example: 1 /service/ip/gprs 123456789 0 Derivation: Mandatory.
ITEM_POID	X(255)	POID of the item object affected due to this event. Applies only to the balance array element that impacts currency balance elements. This might be different from the PIN_FLD_ITEM_OBJ field in the base **/event** class. Example: 1 /item/misc 123456789 0 Derivation: Mandatory.
ORIGINAL_EVENT_POID	X(255)	POID of the original recorded event. Set only if the event has been extracted for pipeline rerating. Example: "1 /event/delayed 123456 0" Derivation: Optional.

Table 79-22 (Cont.) Associated BRM Balance Record Fields

Name	Format	Description
PIN_TAX_LOCALES	X(255)	Used for tax calculation. Values: "*order_origin*|*order_accept*|*ship_from*|*ship_to*" (Note that these fields are separated by pipes (|).) Each of these values (*order_origin*, *order_accept*, *ship_from*, *ship_to*) is an address in the following format: *city*;*zipcode*;*state*;*country*; [*geocode*,*location_mode*,*international_indicator*] **Note:** Be aware of the semicolon separators and enclosing brackets. For example, "cupertino;95014;CA;US; [5723121,2,0]"Derivation: Optional. *order_origin*, *order_accept*, and *ship_from* addresses are all the same and are derived from account profile object tax supplier information. *ship_to* is the address in the first element of the account's NAMEINFO array. *geocode* is either a geocode or NPA-NXX (the first 6 digits of the phone number). *location_mode* is 1 if it is a geocode and 2 if its NPA-NXX. *international_indicator* is 0 (US) or 1 (International). **Important:** This field might not be implemented in this release.
PIN_TAX_SUPPLIER_ID	X(255)	POID of the **/profile/tax_supplier** object used to tax this event. NULL if there is no tax supplier specified. Derivation: Optional. **Important:** This field might not be implemented in this release.
PIN_PROVIDER_ID	X(255)	POID of the remittance service provider account. Example: 1 /account 123456789 0 Derivation: Optional. **Important:** This field might not be implemented in this release.
PIN_INVOICE_DATA	X(255)	Stores the data in the event that is include in the invoice. T The data is mapped to BRM fields and is stored as a string in the format: "@INTEGRATE#PIN_FLD_CALLING_NUMBER#PIN_FLD_CALLED_NUMBER#PIN_FLD_SVC_TYPE# PIN_FLD_SVC_CODE#PIN_FLD_NUMBER_OF_UNITS#PIN_FLD_USAGE_CLASS#PIN_FLD_DNIS#PIN_FLD_BAL_IMPACTS" Arrays follow this format: PIN_FLD_BAL_IMPACTS = *ID,PIN_FLD_RATE_TAG,PIN_FLD_QUANTITY>* For example: @INTEGRATE#004917165210#0049171235292#T1#11#0#USAGE#GSMThing#<1 /item 3456 1#1234,10.0#rateTag#1.000000#|1 /item 5678 1#6789#20.0#rateTag#1.000000>

Table 79-22 (Cont.) Associated BRM Balance Record Fields

Name	Format	Description
NUMBER_OF_BALANCE_IMPACT_PACKETS	9(2)	Specifies the number of packets following these base fields (dynamic structure); for example, 03 means that 3 packets are following. Must be used to evaluate how the record structure continues. Values: 00 - 99: optionally, 0..n packets might follow Derivation: Mandatory.

Supplementary Balance Impact Packet Record (RECType 600)

The packets can optionally be used to store an event-related balance impact array. Within this structure, N-times PIN_BALANCE_IMPACTs are created, each containing one balance impact per RESOURCE_ID and optionally per GL_ID.

This record is used for evaluation event-related balance impacts together with the REL to map rating-internal Charge-Packets to BRM related balance impacts.

- **Condition**: Only relevant if present. If present, a mapping to all PIN-related values has to take place.
- **Derivation**: Optional. From the BRM object **/event/PIN_FLD_BAL_IMPACTS**. Will be optionally generated by a post-processor. If not present, the mapping will take place within the Rated Event (RE) Loader.

  Table 79-23 describes the Supplementary Balance Impact Packet Record fields.

Table 79-23 Supplementary Balance Impact Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 600
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
ACCOUNT_POID	String	POID of the account that the balance impact applies to. Derivation: Mandatory. Calculated.
BAL_GRP_POID	String	Balance group that the balance impact applies to. Derivation: Mandatory. Calculated.
OBJECT_CACHE_TYPE	Integer	Mandatory. Calculated.

Table 79-23 (Cont.) Supplementary Balance Impact Packet Record Fields

Name	Format	Description
ITEM_POID	String	POID of the item that the balance impact applies to. Derivation: Mandatory. Calculated.
PIN_RESOURCE_ID	9(9)	Numeric value of the balance element that is impacted; for example, 840 for US dollars. Values: Any configured BRM balance element ID. Derivation: Mandatory.
PIN_RESOURCE_ID_ORIG	Integer	Optional.
PIN_IMPACT_CATEGORY	X(255)	Name of the BRM impact category that was used to generate this balance impact for the rated event. Values: Any configured BRM impact category. Derivation: Mandatory.
PIN_IMPACT_TYPE	Integer	Mandatory. Calculated.
PIN_GL_ID	9(9)	GLID associated with this balance impact. Values: Any configured BRM general ledger ID. Derivation: Optional, default 0. Derived from IFW_RATEPLAN_CNF.GLACCOUNT. Might be mapped from the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_GL_ID**.
RUM_ID	Integer	Mandatory. Calculated. Default = 0.
PIN_OFFERING_POID	String	Optional. Calculated.
PIN_TAX_CODE	X(255)	Tax code for the rate that was used. When taxes do not apply, this field is set to 0. Derivation: Optional. From IFW_RATEPLAN_CNF.GLACCOUNT->TAXCODE. Might be mapped from the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_GL_ID**.
PIN_RATE_TAG	X(255)	Description of the rate used. Same as the PIN_FLD_DESCR in **/rate**. Can be used to more precisely describe the balance impact detail; for example, the following concatenated, comma-separated rating-related Charge-Packet values used: TIMEZONE, DAY_CODE, TIME_INTERVAL. Values: Free defined text value. Derivation: Optional, default empty. From the BRM object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_RATE_TAG**. The post-mapping processor might decide what mapping-rule applies to this attribute.

Table 79-23 (Cont.) Supplementary Balance Impact Packet Record Fields

Name	Format	Description
PIN_LINEAGE	X(255)	Lineages of event fields if zone map is used in charge selection. Can be used to more precisely describe the balance impact detail; for example, the following concatenated, comma-separated rating-related Charge-Packet values used: ZONEMODEL, SERVICE_CODE, SERVICE_CLASS, IMPACT_CATEGORY,.RESOURCE, RUMGROUP, PRICEMODEL. Values: Free defined text value. Derivation: Optional, default empty. From the BRM object **/event/ PIN_FLD_BAL_IMPACTS.PIN_FLD_LINEAGE**. The post-mapping processor might decide what mapping-rule applies to this attribute.
PIN_NODE_LOCATION	X(255)	Lineage information for the charge offer. See description in charge offers array of **/account**. Can be used to more precisely describe the balance impact detail; for example, the following concatenated, comma-separated rating-related Charge-Packet values used: REVENUEGROUP, DISCOUNTMODEL. Values: Free defined text value. Derivation: Optional, default empty. From the BRM object **/event/ PIN_FLD_BAL_IMPACTS.PIN_FLD_NODE_LOCATION**. The post-mapping processor might decide what mapping-rule applies to this attribute.
PIN_QUANTITY	9(15)	Charged quantity value (beats, duration, volume), as calculated via the related RATEPLAN. Contains the rounded quantity value as it has been calculated during rating. Values: Maximum: 999999999999999 **Note:** In case of Multiple-RUM rating, this value might not be totalizable because different UoMs can logically not be aggregated. In this case, the value is set to 0.Derivation: Optional, default 0. From the BRM object **/event/ PIN_FLD_BAL_IMPACTS.PIN_FLD_QUANTITY**. The post-mapping processor might decide what mapping-rule applies to this attribute; for example, multiple charge packet values might be totalized.

Table 79-23 (Cont.) Supplementary Balance Impact Packet Record Fields

Name	Format	Description
PIN_AMOUNT	9(11)	Amount of impact for one balance element to the account balance. The value might be either positive or negative. The value is added to the PIN_FLD_CURRENT_BAL field of the PIN_FLD_BALANCES array in the account object specified by PIN_FLD_ACCOUNT_OBJ. **Note:** In case of Multiple-RUM rating, this value might be a totalized value. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory. From the BRM object **/event/ PIN_FLD_BAL_IMPACTS.PIN_FLD_AMOUNT**. The post-mapping processor might decide what mapping-rule applies to this attribute; for example, multiple charge packet values might be totalized. **Note:** This value does not include any granted discounts.
PIN_AMOUNT_ORIG	Decimal	Optional.
PIN_PERCENT	Decimal	Optional
PIN_AMOUNT_DEFERRED	Decimal	Optional. Calculated.
PIN_DISCOUNT	9(11)	The discount applied to this balance impact. Can be used to determine the total charge amount value. Note: The AMOUNT value never contains this DISCOUNT value. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory, default 0.
PIN_INFO_STRING	X(2000)	Stores the pricing type.

Supplementary Sub-Balance Impact Packet Record (RECType 605)

Stores balance impacts for sub-balances.

Table 79-24 describes the fields in the Supplementary Sub-Balance Impact Packet Record.

Table 79-24 Supplementary Sub-Balance Impact Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 605 Mandatory.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
BAL_GRP_POID	String	Balance group that the balance impact applies to. Derivation: Mandatory. Calculated.
PIN_RESOURCE_ID	9(9)	Numeric value of the balance element that is impacted; for example, 840 for US dollars. Values: Any configured BRM balance element ID. Derivation: Mandatory.
NEXT_BAL	Decimal	None.
DELAYED_BAL	Decimal	None.
GRANTOR	String	The charge offer or discount offer that granted this balance element.
VALID_FROM_DETAILS	Integer	Sub-balance start time mode (such as first-usage or relative) and relative offset and unit.
VALID_TO_DETAILS	Integer	Sub-balance end time mode (such as relative) and relative offset and unit.

Supplementary Sub-Balance Info Packet Record (RECType 607)

Stores validity dates for sub-balances.

Table 79-25 lists the fields in the Supplementary Sub-Balance Info Packet Record.

Table 79-25 Supplementary Sub-Balance Info Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 607 Mandatory.

Table 79-25 (Cont.) Supplementary Sub-Balance Info Packet Record Fields

Name	Format	Description
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
PIN_AMOUNT	Decimal	Sub-balance amount.
VALID_FROM	Date	Valid from date for this sub-balance.
VALID_TO	Date	Valid to date for this sub-balance.

Tax Jurisdiction Packet

Table 79-26 lists the fields in the Tax Jurisdiction Packet.

Table 79-26 Tax Jurisdiction Packet Fields

Name	Format
RECORD_TYPE	String
RECORD_NUMBER	Integer
PIN_TAX_TYPE	String
PIN_TAX_VALUE	Decimal
PIN_AMOUNT	Decimal
PIN_TAX_RATE	String
PIN_AMOUNT_GROSS	Decimal

EDR Container Fields for Balance Monitoring

The following fields are used for handling balance monitor information.

MONITOR_LIST (DETAIL.CUST_A.ML)

The MONITOR_LIST packet contains information about the balance monitor.

Table 79-27 lists the fields in the MONITOR_LIST packet.

Table 79-27 MONITOR_LIST Packet Fields

Name	Format	Description
BALANCE_GROUP_ID	String	Balance monitor group ID. Mandatory.
MONITOR_OWNER_ACCT_ID	String	Monitor owner's account ID. Mandatory.

Table 79-27 (Cont.) MONITOR_LIST Packet Fields

Name	Format	Description
MONITOR_OWNER_ID	String	Monitor owner ID. Mandatory.
MONITOR_OWNER_TYPE	String	Monitor owner type. Mandatory.

MONITOR_PACKET (DETAIL.ASS_PIN.MP)

The MONITOR_PACKET packet stores information about the balance monitor impacts. This information is added to the Associated Billing Record to be loaded into the database.

Table 79-28 lists the fields in the MONITOR_PACKET packet.

Table 79-28 MONITOR_PACKET (DETAIL.ASS_PIN.MP) Fields

Name	Format	Description
RECORD_TYPE	String	Type of record. Extended to be 3 bytes long. Possible value: 800
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. Important: This record number can change if the sequence of records changes; for example, if new record types are inserted.
ACCOUNT_POID	String	POID of the account that the monitor balance impact applies to. Derivation: Mandatory. Calculated.
BAL_GRP_POID	String	Balance monitor group that the monitor balance impact applies to. Derivation: Mandatory. Calculated.
PIN_RESOURCE_ID	9(9)	Numeric value of the balance element that is impacted; for example, 840 for US dollars. Possible value: Any configured balance element ID. Derivation: Mandatory.

Table 79-28 (Cont.) MONITOR_PACKET (DETAIL.ASS_PIN.MP) Fields

Name	Format	Description
PIN_AMOUNT	9(11)	Amount of impact for one balance element to the monitor balance. The value might be either positive or negative. The value is added to the PIN_FLD_CURRENT_BAL field of the PIN_FLD_BALANCES array in the account's monitor object specified by PIN_FLD_ACCOUNT_OBJ field. **Note:** In case of Multiple-RUM rating, this value might be a total value. Possible values: Price (see below for maximum and minimum). If no price given, space; for example, NULL in a database. The format is variable floating point. The floating decimal point must be set if the given value is not in the required format. Example: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory. Derived from the object **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_AMOUNT**. The post-mapping processor decides what mapping rule applies to this attribute; for example, add multiple charge packet values. **Note:** This value does not include any granted discounts.

MONITOR_SUB_BAL_IMPACT (DETAIL.ASS_PIN.MSBI)

Table 79-29 lists the fields in the MONITOR_SUB_BAL_IMPACT packet.

Table 79-29 MONITOR_SUB_BAL_IMPACT Fields

Name	Format	Description
RECORD_TYPE	String	Type of record. Extended to be 3 bytes long. Possible value: 805
RECORD_NUMBER	9(9)	Sequence number of the record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** This record number can change if the sequence of records changes; for example, if new record types are inserted.
BAL_GRP_POID	String	Balance monitor group that the monitor balance impact applies to. Derivation: Mandatory. Calculated.

Table 79-29 (Cont.) MONITOR_SUB_BAL_IMPACT Fields

Name	Format	Description
PIN_RESOURCE_ID	9(9)	Numeric value of the balance element that is impacted; for example, 840 for US dollars. Possible values: Any configured balance element ID. Derivation: Mandatory.
MONITOR_SUB_BAL	SB	Sub-balance monitor.

MONITOR_SUB_BAL (DETAIL.ASS_PIN.MSB)

Table 79-30 lists the MONITOR_SUB_BAL Packet fields.

Table 79-30 MONITOR_SUB_BAL Fields

Name	Format	Description
RECORD_TYPE	String	Type of record. Extended to be 3 bytes long. Possible value: 807 Derivation: Mandatory.
RECORD_NUMBER	Integer	Contains the UTC time offset that normalizes the VALID_FROM timestamp to the UTC time zone.
PIN_AMOUNT	Decimal	Sub-balance amount.
VALID_FROM	Date	Contains a timestamp of the event end time, rounded to midnight.
VALID_TO	Date	Contains a timestamp of the VALID_FROM time plus 1 day.
CONTRIBUTOR	String	None
NEXT_BAL	Decimal	None
DELAYED_BAL	Decimal	None
ACCOUNT_POID_STR	String	None
SERVICE_POID_STR	String	None
OFFERING_POID_STR	String	None
START_T	Date	None
UTC_TIME_OFFSET	String	None
DESCRIPTION	String	Optional.
FLAGS	Integer	Optional.

Associated Invoice Data Record (RECType @INTEGRATE)

The Associated Invoice Data Record stores data for displaying on invoices.

Table 79-31 lists the fields in the Associated Invoice Data Record.

Table 79-31 Associated Invoice Data Record Fields

Name	Format	Description
RECORD_TYPE	String	The name of the invoice data template, preceded by the @ symbol. Values: @INTEGRATE
A_NUMBER	String	See A_NUMBER.
B_NUMBER	String	See B_NUMBER.
BASIC_SERVICE	String	See BASIC_SERVICE.
NUMBER_OF_UNITS	Decimal	See NUMBER_OF_UNITS.
USAGE_CLASS	String	See USAGE_CLASS.
TERMINATING_SWITCH_IDENTIFICATION	String	See TERMINATING_SWITCH_IDENTIFICATION.
BALANCE_IMPACT	N/A	Balance impact data.
INVOICE_DATA_TERMINATOR	String	N/A

Associated Zone Breakdown Record (RECType 960-969)

Stores zoning information. For each evaluated zone type, a single Zone Breakdown Record is generated, following the Basic Detail Record (020, 021, 030, 031, etc.). A new Basic Record or the Trailer Record end the sequence of Zone Breakdown Records. Also for zone values already contained within the Basic Detail Record, these sub-details could be generated.

Table 79-32 lists the fields in the Associated Zone Breakdown Record.

Table 79-32 Associated Zone Breakdown Record Fields

Name	Format	Description
RECORD_LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long. Values: 960: Standard Zoning (multiple global Zoning per logical EDR Format) 961: Segmentation Zoning (multiple Zoning per customer segment)
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.

Table 79-32 (Cont.) Associated Zone Breakdown Record Fields

Name	Format	Description
CONTRACT_CODE	X(20)	External unique contract code as defined within the associated billing system. Uniquely identifies a charge offer-related contract. Could by used by any post-processors to look up and reference contract, subscriber, and customer data (if needed later on within this post processor). Derivation: Optional. As assigned to an ACCOUNT Object and referenced by a primary CLI. Alternatively the A Number could be used as reference.
SEGMENT_CODE	X(5)	External Segmentation ID as defined within the associated billing system or as defined within the rating process. Segments could vertically group multiple subscriber (for example, for quality reasons) or network operator. Could be used by any post-processor to identify the related customer/network segment that was used during the rating processor for this A Number. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a SUBSCRIBER Object related to the A Number or as assigned to a network operator related to the file stream.
CUSTOMER_CODE	X(20)	External Customer Code as defined within the associated billing system. Could group multiple subscribers. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Optional. As assigned to a CUSTOMER Object and referenced by a primary CLI.
ACCOUNT_CODE	X(20)	External Customer-Account Code as defined within the associated billing system. Could group multiple charge offers assigned to a customer. A customer might have multiple accounts. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Mandatory. As assigned to a CUSTOMER Object and referenced by a primary CLI.
SYSTEM_BRAND_CODE	X(5)	External system or brand, specialist system Code as defined within the associated rating or billing. Could be used for vendor-specific reasons (for example, reseller code or target system identification for post processing, NOSP identification, etc.). Derivation: Mandatory, default 0. As defined within the SYSTEM_BRAND Object and assigned to a PRODUCT Object referenced by a primary CLI.

Table 79-32 (Cont.) Associated Zone Breakdown Record Fields

Name	Format	Description
SERVICE_CODE	X(5)	Internal (mapped, normalized) Service Code used for the zone determination within the associated rating or billing processor, out of the object IFW_SERVICE (.CODE). Could be used by any post-processor to evaluate the service that was really used during the related rating process. Derivation: Mandatory. The external service code is mapped to a unique representation, either: • Out of the service code included in the origin record (might be mapped). • Out of the service code associated to the SUBSCRIBER's A Number.
CUSTOMER_RATEPLAN_CODE	X(10)	The Original charge offer related and Customer/Subscriber specific charge as defined within the associated billing system. If no customer data is present, the actual, internally used charge could be used instead. Could be used by any post-processor to evaluate the charge that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated charge.
SLA_CODE	X(5)	The Original charge offer-related and customer-specific Service Level Agreement as defined within the associated billing system. If no customer data is present, the actual, internally default value could be used instead. Could be used by any post-processor to evaluate the charge that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated charge.
CUSTOMER_BILLCYCLE	X(2)	The Customers associated Billcycle Code as defined within the associated billing system. Could be used by any post-processor to evaluate the billcycle period that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
CUSTOMER_CURRENCY	X(3)	The Customers associated Currency as defined within the associated billing system. Could be used by any post-processor to evaluate the currency and to apply exchange rates that apply to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.

Table 79-32 (Cont.) Associated Zone Breakdown Record Fields

Name	Format	Description
CUSTOMER_TAX_GROUP	X(5)	The Customers associated Tax Group Code as defined within the associated billing system. Could be used by any post-processor to evaluate the tax rate (together with the charge configuration related G/L account's tax code) that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
NUMBER_OF_ZONE_PACKET	9(2)	Defines the number of supplementary zone records following these base fields (dynamic structure); for example, a number of '05' means that 5 records are following. Must be used to evaluate how the record structure continues. Values: 01 - 99: A minimum of at least 1 record is required Derivation: Mandatory.

Supplementary Zone Packet Record (RECType 660)

For each zone model (evaluated by any rating processor), one packet is added to this structure.

This applies only to standard and segmentation zoning (where multiple zoning is possible). All other zone values are listed in the charge breakdown records.

Table 79-33 lists the fields in the Supplementary Zone Packet Record.

Table 79-33 Supplementary Zone Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Extended to be 3 bytes long. Value: 660: Zone Packet Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.

Table 79-33 (Cont.) Supplementary Zone Packet Record Fields

Name	Format	Description
ZONEMODEL_CODE	X(10)	External Zone Model Code as defined in the related ZONEMODEL Object (.CODE) of the rating processor. Could be used by any post-processor to evaluate the zone model that was used during the related rating process. Derivation: Mandatory.
ZONE_RESULT_VALUE_WS	X(5)	Wholesale zone result value as defined for the zone model references by the field ZONEMODEL_CODE. Could be used by any post-processor to evaluate the wholesale zone value that was estimated during the related rating process. Derivation: Optional.
ZONE_RESULT_VALUE_RT	X(5)	Retail zone result value as defined for the zone model references by the field ZONEMODEL_CODE. Could be used by any post-processor to evaluate the retail zone value that was estimated during the related rating process. Derivation: Mandatory.
ZONE_ENTRY_NAME	String	Calculated, will be used by zoning and rating modules.
ZONE_DESCRIPTION	String	Calculated, will be used by zoning and rating modules.
DISTANCE	9(5)	Distance value as calculated by any geographical zone model. Could be used by any post-processor to evaluate the distance value that was estimated during the related rating process. Condition: This applies only if the associated zone model references to a geographical one. Values: The value is given in full and rounded kilometers; for example, 00150 for 150 km. Derivation: Optional. Dependent on the setup of the zone models within the related rating processor. This value represents the internal calculated distance.

Associated Charge Breakdown Record (RECType 970-998)

Stores charge data. For each evaluated charge or partial charge, a single Charge Breakdown Record might be generated, following the Basic Detail Record. A new Detail Record or the Trailer Record end the sequence of Charge Breakdown Records. For charge values already contained within the Basic Detail Record, these sub-details could be generated.

Table 79-34 lists the fields in the Associated Charge Breakdown Record.

Table 79-34 Associated Charge Breakdown Record Fields

Name	Format	Description
RECORD_LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long. Values: 980: Global Charge (multiple EDR-format-related charge) 981: Customer Charge (subscriber-related charge) 982: Reseller/SP Charge (specialist-system-related charge) 983: Content Provider Charge (content-related charge) 990: Carrier Interconnection Charge (trunk-related charge) 991: Reseller Interconnection Charge (EDR-format-related charge) Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
CONTRACT_CODE	X(20)	External unique contract code as defined within the associated billing system. Uniquely identifies a charge offer-related contract. Could by used by any post-processors to look up and reference contract, subscriber, and customer data (if needed later on within this post-processor). Derivation: Optional. As assigned to an ACCOUNT Object and referenced by a primary CLI. Alternatively the A Number could be used as reference.
SEGMENT_CODE	X(5)	External Segmentation ID as defined within the associated billing system or as defined within the rating process. Segments could vertically group multiple subscriber (for example, for quality reasons) or network operator. Could be used by any post-processor to identify the related customer/network segment that was used during the rating processor for this A Number. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a SUBSCRIBER Object related to the A Number or as assigned to a network operator related to the file stream.

Table 79-34 (Cont.) Associated Charge Breakdown Record Fields

Name	Format	Description
CUSTOMER_CODE	X(20)	External Customer Code as defined within the associated billing system. Could group multiple subscribers. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Optional. As assigned to a CUSTOMER Object and referenced by a primary CLI.
ACCOUNT_CODE	X(20)	External Customer-Account Code as defined within the associated billing system. Could group multiple charge offers assigned to a customer. A customer might have multiple accounts. Could be used by any post-processors as an alternative identifier to look up and reference subscriber or customer data (if needed later on within this post-processor). Derivation: Mandatory. As assigned to a CUSTOMER Object and referenced by a primary CLI.
SYSTEM_BRAND_CODE	X(5)	External system or brand, specialist system Code as defined within the associated rating or billing. Could be used for vendor-specific reasons (for example, reseller code or target system identification for post-processing, NOSP identification, etc.) Derivation: Mandatory, default 0. As defined within the SYSTEM_BRAND Object and assigned to a PRODUCT Object referenced by a primary CLI.
SERVICE_CODE	X(5)	Internal (mapped, normalized) Service Code used for the zone determination within the associated rating or billing processor, out of the object IFW_SERVICE (.CODE). Could be used by any post-processor to evaluate the service that was really used during the related rating process. Derivation: Mandatory. The external service code is mapped to a unique representation, either: • Out of the service code included in the origin record (might be mapped). • Out of the service code associated to the SUBSCRIBERs A Number.

Table 79-34 (Cont.) Associated Charge Breakdown Record Fields

Name	Format	Description
CUSTOMER_RATEPLAN_CODE	X(10)	The Original charge offer related and Customer/Subscriber specific charge as defined within the associated billing system. If no customer data is present, the actual, internally used charge could be used instead. Could be used by any post-processor to evaluate the charge that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated charge.
SLA_CODE	X(5)	The Original charge offer-related and customer-specific Service Level Agreement as defined within the associated billing system. If no customer data is present, the actual, internally default value could be used instead. Could be used by any post-processor to evaluate the charge that was really used during the related rating process. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to an ACCOUNT Object and referenced by a primary CLI or as assigned to the associated charge
CUSTOMER_BILLCYCLE	X(2)	The Customer's associated Billcycle Code as defined within the associated billing system. Could be used by any post-processor to evaluate the billcycle period that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI
CUSTOMER_CURRENCY	X(3)	The Customer's associated Currency as defined within the associated billing system. Could be used by any post-processor to evaluate the currency and to apply exchange rates that apply to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.
CUSTOMER_TAXGROUP	X(5)	The Customers associated Tax Group Code as defined within the associated billing system. Could be used by any post-processor to evaluate the tax rate (together with the charge configuration-related G/L account's tax code) that applies to this call. Derivation: Only mandatory for RECORD_TYPE 981, 984. As assigned to a CUSTOMER Object and referenced by a primary CLI.

Table 79-34 (Cont.) Associated Charge Breakdown Record Fields

Name	Format	Description
NUMBER_OF_CHARGE_PACKETS	9(2)	Defines the number of CBRs (charge breakdown records); does not reflect the actual number of charge packets per CBR.
NUMBER_OF_TAX_PACKETS	Integer	Mandatory. Calculated. Default = 1.
CUSTOMER_OPENING_BALANCE	Decimal	If prepaid rated call, the opening balance for the subscriber. Optional.
CUSTOMER_CLOSING_BALANCE	Decimal	If prepaid rated call, the closing balance for the subscriber. Optional.
RUM_NAME	String	Optional. Calculated.
FU_DISCOUNT_OBJECTS	String	The account's discounts that have first-usage start times which were used to discount the event. Mandatory. Calculated.

Update Balance Packet

This block contains initialized sub-balances of related balance elements based on a bundle.

Table 79-35 lists the fields in the Update Balance Packet.

Table 79-35 Update Balance Packet Fields

Name	Format	Description
RECORD_TYPE	String	The type of call record. Mandatory.
BALANCE_GROUP_ID	Integer	POID of the account's balance group for which a balance element balance starts on first usage. Mandatory.
RESOURCE_ID	Integer	ID of the associated balance element. Mandatory.
RECORD_NUMBER	Integer	Sequence number of the record in the file. Mandatory.
VALID_FROM	Date	The balance element balance start time. Mandatory.
VALID_TO	Date	The balance element balance end time. Mandatory.
VALID_FROM_DETAIL	Integer	The start time mode (such as first-usage or relative), relative offset unit (such as minutes, months, or cycles), and number of offset units. Mandatory.

Table 79-35 (Cont.) Update Balance Packet Fields

Name	Format	Description
VALID_TO_DETAIL	Integer	The end time mode (such as relative), relative offset unit (such as minutes, months, or cycles), and number of offset units. Mandatory.
CONTRIBUTOR	String	Balance group contributor.
GRANTOR	String	Balance group grantor.
GRANT_VALID_FROM	Date	Grant validity start time.
GRANT_VALID_TO	Date	Grant validity end time.

RUM Map Block

RUM_MAP block contains all the RUMs that are used in the ACB block.

Table 79-36 lists the fields in the RUM Map Block.

Table 79-36 RUM Map Block Fields

Name	Format	Description
RECORD_TYPE	String	None
RECORD_NUMBER	Integer	None
RUM_NAME	String	None
NET_QUANTITY	Decimal	Contains the summation of BALANCE_PACKET.PIN_QUANTITY for RUM_NAME.
UNRATED_QUANTITY	Decimal	None

Supplementary Minimum Charge Information

Minimum charge information (the MINIMUM_CHARGE block) prevents charging a customer less than the minimum charge for a call. The values are taken from the pricing configuration.

Table 79-37 lists the Supplementary Minimum Charge Information fields.

Table 79-37 Supplementary Minimum Charge Information Fields

Name	Format	Description
RESOURCE	String	Balance Element used when rating the call.
TOTAL_CHARGE_VALUE	Decimal	Total charge of the call.
MINIMUM_CHARGE_VALUE	Decimal	Minimum charge of the call.

Supplementary Charge Packet Record (RECType 660)

For each charge (evaluated by any rating processor), at least one packet is added to this structure. This applies only to rating models as defined in the record type.

> **Note:**
>
> If a call had to be split over several time zones, there is a separate packet for each part of the call. The charge sum of all parts (where the related charge is equal) represents the total charge of the related basic detail EDR.

Table 79-38 lists the fields in the Supplementary Charge Packet Record.

Table 79-38 Supplementary Charge Packet Record Fields

Name	Format	Description
RECORD_TYPE	String	Value: 660: Charge Packet Record Derivation: Mandatory. Set by the first processor and left unchanged.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. **Important:** Following modules might change this record number; for example, if new record types are inserted.
RATEPLAN_CODE	X(10)	External Charge Code as defined in the related RATEPLAN Object (.CODE) used by the rating process. Could be used by any post-processor to evaluate the RATEPLAN that was used during the related rating process. Derivation: Mandatory.
RATEPLAN_TYPE	X(1)	A charge could be either wholesale or retail. Could be used by any post-processor to determine if the charge calculated throughout the RATEPLAN used is a wholesale or retail one. Values: W: Wholesale R: Retail Derivation: Mandatory. Taken from the setup related to the RATEPLAN_CODE Field
RATETAG_CODE	String	Derivation: Mandatory. Calculated.
ZONEMODEL_CODE	X(10)	External Zone Model Code as defined in the related ZONEMODEL Object (.CODE) defined within the related charge used by the rating process. Could be used by any post-processor to evaluate the zone model that was really used during the related rating process. Derivation: Mandatory.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
SERVICE_CODE_USED	X(5)	Internal (RATEPLAN related mapped) Service Code used for the zone determination within the associated rating or billing processor, out of the object IFW_RATESERVICE_MAP (.NEW_SERVICECODE). Could be used by any post-processor to evaluate the service that was really used during the related rating process. Derivation: Mandatory.
SERVICE_CLASS_USED	X(5)	External Service Class Used as defined within the RATEPLAN (for example, for specific QoS) and used by the rating process. Could be used by any post-processor to evaluate the level of service quality that was really used during the related rating process. Derivation: Mandatory. Default = '*'.
IMPACT_CATEGORY	X(10)	Impact Category result value as defined; for example, a zone value references or a usage scenario map result. Could be used by any post-processor to evaluate the zone value that was estimated during the related rating process. Derivation: Mandatory.
ZONE_DESCRIPTION	String	Calculated. Used by zoning and rating modules.
IC_DESCRIPTION	String	Calculated. Used by zoning and rating modules.
ZONE_ENTRY_NAME	String	Calculated. Used by zoning and rating modules.
DISTANCE	9(5)	Distance value as calculated by any geographical zone model. Could be used by any post-processor to evaluate the distance value that was estimated during the related rating process. Condition: This applies only if the associated zone model references to a geographical one. Values: The value is given in full and rounded kilometers; for example, 00150 for 150 km. Derivation: Optional. Dependent on the setup of the zone models within the related rating processor. This value represents the internal calculated distance.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
TIMEMODEL_CODE	X(10)	External Time Model Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). Could be used by any post-processor to evaluate the time model that was really used during the related rating process. Derivation: Mandatory. The time model is given by evaluating the RATEPLAN configuration and the starting timestamp of the record.
TIMEZONE_CODE	X(10)	External Time Zone Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). Could be used by any post-processor to evaluate the time zone that was really used during the related rating process. Derivation: Mandatory. The time zone is given by evaluating the RATEPLAN configuration and the starting timestamp of the record within the related TIMEMODEL.
DAY_CODE	X(10)	External Day Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). This attribute describes the evaluated day code within the above relationship. Could be used by any post-processor to evaluate the day code that was really used during the related rating process, even if single charge packets are being generated in case of time zone splitting. Derivation: Mandatory. DAY_CODE as defined in the related DAYCODE object (.CODE). The day is given by evaluating the RATEPLAN configuration and the starting timestamp of the record within the related TIMEMODEL and TIMEZONE. If Splitting of single charge packets is not performed, the day code of the start of the call is being used.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
TIME_INTERVAL_CODE	X(10)	External Time Interval Code as estimated and used by the rating process. Time model and Time zone define a unique relationship between a day code (special day, weekday, weekend, etc.) and a time interval (time band within a day). This attribute describes the evaluated time interval within the above relationship. Could be used by any post-processor to evaluate the time interval that was really used during the related rating process, even if single charge packets are being generated in case of time zone splitting. Derivation: Mandatory. TIME_INTERVAL Code as defined in the related TIME_INTERVAL object (.CODE). The time zone is given by evaluating the RATEPLAN configuration and the starting timestamp of the record within the related TIMEMODEL and TIMEZONE. If Splitting of single charge packets is not performed, the time interval of the start of the call is being used.
PRICEMODEL_CODE	X(10)	External Pricing Code as defined in the related PRICEMODEL Object (.CODE) used by the rating process. Could be used by any post-processor to evaluate the pricing that was really used during the related rating process. Derivation: Mandatory. Dependent on the setup of the RATEPLAN within the related rating processor. This value represents the external pricing code as it had been set up and used.
PRICEMODEL_TYPE	X(1)	Defines which type of the pricing was used for this charge packet. Could be used by any post-processor to evaluate which pricing was really used during the related rating process. Values: S: Standard pricing was used A: Alternative pricing was used Derivation: Mandatory, default 'S'. Dependent on the setup of the RATEPLAN within the related rating processor. This value represents the external type of pricing as it had been set up and used. A packet with a pricing type 'S' does always exist. If an alternative pricing is configured, a second packet of type 'A' is generated.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
RESOURCE	X(10)	Balance Element, which has been used for rating or discounting purposes. A balance element might be a currency or any other type (for example, loyalty points) that should be used to calculate parallel charges. Could be used by any post-processor to classify the different charge items. Values: Any configured values of the IFW_RESOURCE object. Derivation: Mandatory, directly taken out of the PRICEMODEL configuration appropriate to the charge packet.
RESOURCE_ID	Integer	Derivation: Optional. Calculated. If 0, the RESOURCE field is ignored.
RESOURCE_ID_ORIG	Integer	Optional. Used if the exchange rate module is configured.
RUMGROUP	X(10)	Classifies the charging item which was derived from the service. A RUM group defines a list of RUMs that should be used together to define a total charge. For example, 'total' could consist of 'DUR'+'VOL_S'+'VOL_R'. Could be used by any post-processor to classify the different charge items. Values: Any configured values of the IFW_RUMGROUP object. Derivation: Mandatory, directly taken out of the Service object's RUM group definition.
RUM	X(10)	Classifies the charging part of a call in a intercarrier relationship, for interconnection or roaming. Values: Dependent on the setup of the IFW_RUM object. Filled by default with '*' if multiple RUMs are used within one Charge Packet.
NETWORK_OPERATOR_CODE	X(10)	Network Operator Code (or Reseller / Content Provider Code) as defined in the NO Objects (.CODE) of the related rating process. Could be used by any post-processor (especially by interconnection billing) to evaluate the network operator to which the calculated charge belongs. Condition: Network operators can be assigned as follows: • Directly to an EDR-format • Via a trunk identification (carrier/reseller interconnection) • A content provider code via a special number, b# Derivation: Only mandatory for RECORD_TYPE 983, 990, 991. The network operator is given by evaluating the relationship during the estimation process which RATEPLAN should be used.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
NETWORK_OPERATOR_BILLINGTYPE	X(1)	Classifies the Type of the associated network operator involved within this charge. Could be used by any interconnection processor to distinguish between incoming and outgoing charges. Condition: Only applies to interconnection rating. Values: O: Outgoing NO, charges have to be paid to the related NO I: Incoming NO, charges are received by the related NO Derivation: Only mandatory for RECORD_TYPE 983, 990, 991; else default I. The NO billing type is directly related to the network operator setup.
CHARGE_TYPE	X(1)	Classifies the charging part of a call in an intercarrier relationship, for interconnection or roaming. Could be used by any interconnection processor to classify the different charge types. Condition: Only applies to interconnection rating. Values: I: Inroute Charge (only applies for carrier interconnection) O: Outroute Charge (only applies for carrier interconnection) T: Transit Charge (only applies for carrier interconnection) N: Normal Charge (applies for all other charges, default) Derivation: Only mandatory for RECORD_TYPE 983, 988, 990, 991, 995, 996; else default N. The switch/trunk is directly related to the type setup.
TRUNK_USED	X(15)	Trunk ID or MSC which was used to calculate the interconnection charges within this packet. This field contains the internal, virtual or mapped trunk address and not the external one and is related either to the inroute or outroute trunk (see field CHARGE_TYPE as a reference). Could be used by any interconnection processor to classify the different service/charge types. Condition: Only applies to interconnection rating. Derivation: Only mandatory for RECORD_TYPE 990, 991. The trunk ID is directly related to the network model and mapping rules used.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
POI_USED	X(10)	POI In which was used to calculate the interconnection charges within this packet. Could be used by any interconnection processor to classify the different service/charge types. Condition: Only applies to interconnection rating. Derivation: Only mandatory for RECORD_TYPE 990, 991. The POI is directly related to the network model and mapping rules used.
PRODUCTCODE_USED	X(10)	Internal charge offer which was used to calculate the charges within this packet. This field contains the internal, virtual or mapped network service type and is related either to the inroute or outroute trunk (ICPRODUCT). Could be used by any interconnection processor to classify the different service/charge types. Condition: Only applies to interconnection rating. Derivation: Only mandatory for RECORD_TYPE 990, 991. The Charge Offer Code is directly related to the IC Charge Offer Code configuration.
PIN_LOGIN_ALIAS	String	Optional. Calculated.
CHARGING_START_TIMESTAMP	YYYYMMDD HH24MISS	The timestamp used for charging. Format: YYYYMMDDHHMISS; for example, 19990518190357. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGEABLE_QUANTITY_VALUE	9(15)	Original chargeable units (beats, duration), as provided by the sender (for example, network element or any other given input format). Contains the original, not-rounded quantity value. Might be useful by some kind of processors analyzing as how many units the call was originally treated by the sender and/or as many the record was treated during rating. Values: Maximum: 999999999999999 Examples: CHARGABLE_QUANTITY_VALUE = 87 sec. a) if RATEPLAN is defined with a 60sec. beat -> ROUNDED_QUANTITY_VALUE will contain 120sec. b) if RATEPLAN is defined with a 30sec. beat -> ROUNDED_QUANTITY_VALUE will contain 90sec. Derivation: Optional, defaulted by ROUNDED_QUANTITY_VALUE if not provided or present. Set by either the input format or the rating processor generating this packet and left unchanged.
ROUNDED_QUANTITY_VALUE	9(15)	Charged units (beats, duration), as calculated via the related RATEPLAN. Contains the rounded quantity value as it has been calculated during rating. Might be useful by some kind of processors analyzing as how many units the call was originally treated by the sender and/or as many the record was treated during rating. Values: Maximum: 999999999999999 Examples: CHARGABLE_QUANTITY_VALUE = 87 sec. a) if RATEPLAN is defined with a 60sec. beat -> ROUNDED_QUANTITY_VALUE will contain 120sec. b) if RATEPLAN is defined with a 30sec. beat -> ROUNDED_QUANTITY_VALUE will contain 90sec. Derivation: Mandatory, defaulted 0. Set by the rating processor generating this packet and left unchanged.
ROUNDED_QUANTITY_FROM	Decimal	Mandatory. Calculated.
ROUNDED_QUANTITY_TO	Decimal	Mandatory. Calculated.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
ROUNDED_QUANTITY_UoM	X(3)	The Unit of Measurement associated with the Rounded Chargeable Quantity Value. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: As specified in the database model, if a UoM conversion had been carried out; else as defined in the related Basic Detail Record. Derivation: Mandatory, default 'SEC'. Set by the rating processor and left unchanged.
QUANTITY_FROM	Decimal	Charge packet start quantity.
QUANTITY_TO	Decimal	Charge packet end quantity.
EXCHANGE_RATE	9(11)	Contains the exchange rate which has been used to convert the Incoming currency to the internal currency as indicated in the field CHARGED_CURRENCY_TYPE. Can be used to convert the virtual currency SDR (which is used in conjunction of TAP) to internal currencies and convert the Charge back to SDR after Rating. This would be a typical usage for Interconnection Rating. Values: Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Optional, defaulted 00000000001 (=1,00).
EXCHANGE_CURRENCY	X(3)	Currency code as defined for the exchange rate; for example, "DEM" or "EUR". Can be used to interpret the exchange rate to distinguish to which currency the exchange rate was used for. For example, for TAP: the charged amount might be given in SDR currency and the exchange rate will define the rate used to convert into local currency. Derivation: Optional. As related to the exchange rate value used for. Use the three-digit ISO currency code.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGED_CURRENCY_TYPE	X(1)	Indicates which of the available currencies was used to generate the charge packet. Could be used by any post-processor to classify the different charge packets. Values: R: Rating Currency (default) B: Billing Currency H: Home Currency **Note:** In case of currency conversion, where in parallel all three currency models are supported (R, B, and H); there is one charge packet for each currency type. Derivation: Depending on the function module which generated the charge packet, the value is set to one of the values given above. The rating modules set the value to "R", while the ExchangeRate module generates two charge packets (one for the home currency and one for the billing currency). This feature is usually required for interconnection purposes.
CHARGED_AMOUNT_VALUE	9(11)	The charge for the call (could be any kind of price). A monetary amount assigned to the call by any rating processor. This includes any toll charge but does not include any CAMEL invocation fee. In case of interconnection or roaming charges, this is the advice of charge. Can be used by any post-processor to collect multiple charges related to one record. This opens the possibility to keep more then one charge. With this structure there is the possibility to keep all charges related to the record/call (for example, end-customer, wholesale, content provider, reseller, optimized data warehouse etc.). Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory.
CHARGE_REFUND_INDICATOR	Integer	Optional. Charge refund indicator item.
CHARGED_AMOUNT_VALUE_ORIG	Decimal	Optional. Used if the exchange rate module is configured.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGED_AMOUNT_CURRENCY	X(3)	Currency code as defined within the associated RATEPLAN; for example, DEM or EUR. Can be used to interpret the amount value and to distinguish between different currencies (multicurrency support). Any rating or billing processor might convert the different currencies. Derivation: Mandatory. As related to the RATEPLAN used. Use the three-digit ISO currency code.
CHARGED_TAX_TREATMENT	X(1)	Charges might be inclusive or exclusive of tax. Can be used to interpret the amount value and to distinguish between net and gross charges. Values: Y: Tax included in the charge N: Tax not included in the charge (default) Derivation: Mandatory (default N).
CHARGED_TAX_RATE	9(4)	Defines the tax rate applicable to the charge. Because some national legal definitions dictate that the tax rate applicable is determined by the invoice date, there is a possibility that the rate on the invoice might differ from the rate on the transfer. However, the likelihood of this happening is extremely low. Can be used to interpret the amount value and to convert between net and gross charges or to represent customer-specific rates. Values: 0000 through 9999 (2 fixed decimals) Example: 16.00% 1600 Derivation: Optional. As related to the taxation module used (refer to IFW_TAX.TAXRATE).

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGED_TAX_CODE	X(5)	Defines the tax rate applicable to the charge. Because some national legal definitions dictate that the tax rate applicable is determined by the invoice date, there is a possibility that the rate on the invoice might differ from the rate on the transfer. However, the likelihood of this happening is extremely low. Can be used by any billing processor to interpret the amount value and to convert and calculate a customer-specific tax rate. Values: As defined via the reference IFW_GLACCOUNT.TAXCODE. Example: M16 for tax code M16. Derivation: Optional. As related to the RATEPLAN configuration's general ledger account used.
USAGE_GL_ACCOUNT_CODE	X(10)	The General Ledger Code defines a reference applicable to the usage revenue account. Can be used by any billing processor to interpret the amount value and to convert and calculate a customer-specific tax rate or to trigger any account balance bookings within a financial accounting system. Values: As defined in the database object IFW_RATEPLAN_CNF.USG_GLACCOUNT. Example: USG_AIRTEL for account usage revenue airtime. Derivation: Optional. As related to the RATEPLAN used.
REVENUE_GROUP_CODE	X(5)	The Revenue Group Code defines a reference applicable to a specific group of usage revenue. Different usage groups can be used to define split billing rules to be distributed to different customers; for example, airtime is paid by a subscriber and monthly periodic fees are paid by the customer above. Usage: Can be used by any billing processor to interpret a split billing based on different revenue groups. Values: As defined in the database object IFW_RATEPLAN_CNF.REVENUEGROUP. Example: AIR for usage revenue group airtime. Derivation: Optional. As related to the RATEPLAN used.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
DISCOUNTMODEL_CODE	X(10)	External Discount Code as defined in the related DISCOUNTMODEL Object (.CODE) used by the rating process. Could be used by any post-processor to evaluate the discount that was really used during the related rating process. Derivation: Mandatory. Dependent on the setup of the RATEPLAN within the related rating processor. This value represents the external discount code as it had been set up and used.
GRANTED_DISCOUNT_AMOUNT_VALUE	9(11)	The field records the total discount value, which was granted to calculate the correct CHARGED_AMOUNT_VALUE. Can be used by any post-processor analyzing the discounts that had been granted and must be used to calculate the exact charge (with discount included). This will give a good indicator of the discount structure and usage. **Note:** The CHARGED_AMOUNT_VALUE never contains this DISCOUNT_VALUE. Values: Space: No price given, like NULL in a database Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -9999999999 Maximum: 99999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory, default 0. Note: The currency is always the same currency as given in the field CHARGED_AMOUNT_CURRENCY.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
GRANTED_DISCOUNT_QUANTITY_VALUE	9(15)	The total discount quantity value, which was granted to calculate the correct CHARGED_AMOUNT_VALUE; for example, Applied free minutes. Not updated by discounting. Used in interconnect mapping. Can be used by any post-processor analyzing the discounts that had been granted. Could be used to recalculate the original quantity value (without any discount). This will give a good indicator of the discount structure and usage. Condition: Only relevant for quantity (duration) based service discounts. Values: Maximum: 999999999999999 Derivation: Mandatory, default 0. Might be set by any rating or pre-billing processor.
GRANTED_DISCOUNT_QUANTITY_UoM	X(3)	The Unit of Measurement associated with the Granted Discount Quantity Value. Not updated by discounting. Used in interconnect mapping. Can be used to interpret the quantity value, but usually not needed because the quantity itself is sufficient for all rating steps. Values: As specified in the database model, if a UoM conversion had been carried out; else as defined in the related Basic Detail Record. Derivation: Mandatory, default 'SEC'. Set by the rating processor and left unchanged.
DEFERRED_AMOUNT	Decimal	Optional Calculated
PIN_PERCENT	Decimal	Optional Calculated
NUMBER_OF_DISCOUNT_PACKETS	9(2)	Defines the number of supplementary discount packet records following these base fields (dynamic structure); for example, 05 means that 5 records are following. Must be used to evaluate how the record structure continues. Values: 00 - 99: either zero or N record(s) are following Derivation: Mandatory.
VALID_FROM	Date	Optional.
VALID_TO	Date	Optional.
CYCLE_OFFSET	Integer	Optional. Identifies a grant's validity period.

Table 79-38 (Cont.) Supplementary Charge Packet Record Fields

Name	Format	Description
CHARGE_INDEX	Integer	The array index of incoming charge packets, used by the discount pipeline to match existing charges, in case a credit limit check changed the original charges.

Split Charge Packet

FCT_Discount splits charge packets if necessary during prepaid authorization. Each split charge packet represents a segment with a single net rate, including discounts.

Table 79-39 lists the fields in the Split Charge Packet.

Table 79-39 Split Charge Packet Fields

Name	Format	Description
RESOURCE_ID	Integer	Numeric ID of the balance element. Used for filtering in the discount detail. Copied from the original charge packet.
RUM	String	RUM name. Copied from the original charge packet.
QUANTITY_FROM	Decimal	Split charge packet start quantity. Calculated by the module.
QUANTITY_TO	Decimal	Split charge packet end quantity. Calculated by the module.
CHARGED_AMOUNT_VALUE	Decimal	Amount of the charge for this split charge packet. Calculated by the module.
INTERN_PACKET_INDEX	Integer	The index of the split charge packet.
INTERN_SRC_PACKET_INDEX	Integer	The packet index of the charge packet from which this split charge packet was generated.

Supplementary Last Beat Information

Information about the last beat (the LAST_BEAT_INFO block) is mainly used for abnormal call terminations.

Table 79-40 lists the fields in the Supplementary Last Beat Information block fields.

Table 79-40 Supplementary Last Beat Information Fields

Name	Format	Description
LAST_BEAT_QUANTITY	Decimal	The length of a beat. The value of a beat (for example, clicks or bytes) is defined in the pricing and determined by *FCT_MainRating*.
LAST_BEAT_CHARGE	Decimal	The charge for a beat.

Charge Breakdown Record Tax Packet (RECType 660)

Add code to the OutGrammar to store tax information from the EDR container into the output TAP blocks.

Block. *n* times. Optional.

Table 79-41 lists the fields in the Charge Breakdown Record Tax Packet.

Table 79-41 Charge Breakdown Record Tax Packet Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Must be set to 660.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
TAX_CODE	Integer	None
TAX_RATE	String	None
TAX_VALUE	Decimal	None
TAX_PERCENT	Decimal	None
TAX_VALUE_ORIG	Decimal	Optional. Used when exchange rate is configured.
TAX_TYPE	String	None
CHARGE_TYPE	String	None
TAXABLE_AMOUNT	Decimal	None
TAX_QUANTITY	Decimal	None
RELATED_RECORD_NUMBER	Integer	None
RELATED_CHARGE_INFO_ID	Integer	None
CHARGE_INFORMATION_COUNTER	Integer	None
CHARGE_INFORMATION_COUNTER	Integer	None

Associated Message Description Record (RECType 999)

Stores errors that occur in preprocessing. For each error a single Message Description Record is generated, following the Basic Record. A new Basic Record or the Trailer Record end the sequence of Message Description Records.

Table 79-42 lists the fields in the Associated Message Description Record.

Table 79-42 Associated Message Description Record Fields

Name	Format	Description
RECORD_TYPE	String	Value: 999: Message Description Record Derivation: Mandatory. Set by the first processor and left unchanged. Usage: Determination of the different record types.
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor. Important: Following modules might change this record number; for example, if new record types are inserted.
SYSTEM	X(8)	Description or Code of the system which produced this record; for example, Host name or process name.

Table 79-42 (Cont.) Associated Message Description Record Fields

Name	Format	Description
MESSAGE_SEVERITY	X(1)	Severity code for this message. Values: N: Normal (Hint) W: Warning E: Error (could be either a minor, major, or critical error) Derivation: Mandatory. Set by the first processor and left unchanged.
MESSAGE_ID	N(7)	An error code used to cross-reference the call to the relevant description. Values: 0000000 through 9999999 Derivation: Mandatory. Set by the first processor and left unchanged.
MESSAGE_DESCRIPTION	X(50)	Description of the error. It is mandatory but the content is entirely at the discretion of the pre-processor. Derivation: Mandatory. Set by the first processor and left unchanged.

Associated TAP Error Record

Table 79-43 lists the Associated TAP Error Record fields.

Table 79-43 Associated TAP Error Record Fields

Name	Format
RECORD_TYPE	String
RECORD_NUMBER	Integer
ERROR_NAME	String
ERROR_SEVERITY	Integer
TAP3_ERROR_CODE	String
TAP3_ERROR_APPLICATION_TAG	String
TAP3_ERROR_DEPTH	String

Associated SMS Record (RECType 580)

This optional record is used to store SMS call data.

Table 79-44 lists the fields in the Associated SMS Record.

Table 79-44 Associated SMS Record Fields

Name	Format	Description
RECORD_TYPE	String	Record type for Associated SMS Record. Value: 580 Derivation: Mandatory. Set by the first processor.
RECORD_NUMBER	9(9)	Sequence number of record in file. Used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Auto-generated. Set by the first processor. Important: Record number may change; for example, if new record types are inserted.
CONTENT_INDICATOR	X(1)	Indicator as to the contents of the message sent or received. Derivation: Optional.
ORIGINATING SWITCH IDENTIFICATION	X(10)	SMS-C from which the SMS was issued by the A party. Derivation: Optional.
DESTINATION SWITCH IDENTIFICATION	X(10)	SMS-C from which the SMS was issued to the B party. Derivation: Optional.
PROVIDER ID	X(2)	Unique Service Provider Identifier. Derivation: Optional.
SERVICE ID	X(2)	Unique Service ID. Derivation: Optional.
DEVICE NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Derivation: Optional.
PORT NUMBER	X(24)	Identifies the unique subscriber ID; for example, the IMSI number. Derivation: Optional.
DIALED DIGITS	X(40)	The number dialed by the customer when establishing a call or the number to which the call is forwarded or transferred. Derivation: Optional.

Associated MMS Record (RECType 590)

This optional record is used to store MMS call data.

Table 79-45 lists the Associated MMS Record fields.

Table 79-45 Associated MMS Record Fields

Field Name	Data Format	Description
RECORD TYPE	String	Record type for Associated MMS Record. Value: 590 Derivation: Mandatory. Set by the first processor.
RECORD NUMBER	9(9)	Sequence number of record in file. Used to ensure a linear sequence order for all records; for example, as a sorting criteria. Values: Minimum: 000000002 Maximum: 999999998 Derivation: Mandatory. Auto-generated. Set by the first processor. **Important:** Record number may change; for example, if new record types are inserted.
ACCOUNT STATUS TYPE	X(2)	Indicator of the account type from which the message was sent. Derivation: Optional.
PRIORITY	X(2)	Indicator as to the priority of the message; for example, high, medium or low. Derivation: Optional.
MESSAGE CONTENT	X(255)	Content type; for example, image or plain text. Derivation: Optional.
MESSAGE ID	X(16)	Unique message group ID. If the message was sent as part of a group, an indicator as to which group it was sent from. Derivation: Optional.
STATION IDENTIFIER	X(255)	Station from which message was sent. Value: MMS identifier. Derivation: Optional.
FC INDICATOR	X(9)	Indicator as to whether the message was forwarded or copied. Derivation: Optional.

Table 79-45 (Cont.) Associated MMS Record Fields

Field Name	Data Format	Description
CORRELATION ID	X(16)	Correlation ID for the message if part of a group. Derivation: Optional.
DEVICE NUMBER	X(24)	Identifies the equipment used by the subscriber during the call; for example, the International Mobile Equipment Identity number (IMEI). Derivation: Optional.
PORT NUMBER	X(24)	Identifies the unique subscriber ID; for example, the IMSI number. Derivation: Optional.
DIALED DIGITS	X(40)	The number dialed by the customer when establishing a call or the number to which the call is forwarded or transferred. Derivation: Optional.
CELL ID	X(10)	Cell ID of the A party, or the cell from which the call originated. Derivation: Optional.
B CELL ID	X(10)	Cell ID of the B party, or the cell receiving the call. Derivation: Optional.
A TERM CELL ID	X(10)	Cell ID of the A party when the call terminated. Derivation: Optional.

Trailer Record (RECType 090)

Table 79-46 lists the fields in the Trailer Record.

Table 79-46 Trailer Record Fields

Name	Format	Description
RECORD_LENGTH	Integer	Optional for backward compatibility.
RECORD_TYPE	String	Extended to be 3 bytes long, first byte denotes market like GSM, ISDN. Value: 090: Trailer Record Derivation: Mandatory. Set by the first processor and left unchanged. Usage: Determination of the different record types.

Table 79-46 (Cont.) Trailer Record Fields

Name	Format	Description
RECORD_NUMBER	9(9)	Sequence number of record in file. Can be used to ensure a linear sequence order for all records; for example, as a sorting criteria. Derivation: Mandatory. Set by the first processor.
SENDER	X(10)	Identifies the PLMN or physical (network) operator, which is sending the file, used to determine the network, which is the sender of the data. The full list of mobile codes in use is given in MoU TADIG PRD TD. 13: PLMN Naming Conventions. Can be used to determine a unique NOSP_ID together with the RECIPIENT. Can also be used to determine the network operator responsible for the EDR. Derivation: Optional, but should be defaulted if not present on the input side, for example, by own NO-Id, for example, 'DTAG'. Set by the first processor and left unchanged.
RECIPIENT	X(10)	Identifies the PLMN or physical (network) operator to whom the file is being sent, used to determine the network, which is the recipient of the data. The full list of mobile codes in use is given in MoU TADIG PRD TD. 13: PLMN Naming Conventions. Can be used to determine a unique NOSP_ID together with the SENDER. Can also be used to determine the reseller or service provider who is responsible for billing these CDRs. Derivation: Optional, but should be defaulted if not present on the input side, for example, by own NO-Id, for example, 'DTAG'. Set by the first processor and left unchanged.
SEQUENCE_NUMBER	9(6)	Identifies each file sent by the VPLMN or logical sender to a particular HPLMN or logical recipient. It indicates the file number of the specific file type, starting at 1 and increments by one for each new file of that type sent. Separate sequence numbering must be used for Test- and Chargeable-Data. Having reached the maximum value (99999), the number must recycle to 1. **Note:** In the case of retransmission for any reason, this number does not increment. Range: 000001 - 999999 for Test Data 000001 - 999999 for Chargeable Data Derivation: Mandatory. Set by the first processor and could be changed by any following processor in case of recycling to assure a unique and linear sequence order to all following processors.

Table 79-46 (Cont.) Trailer Record Fields

Name	Format	Description
ORIGIN_SEQUENCE_NUMBER	9(6)	Original file sequence number as generated the first time. Identical content as for the SEQUENCE_NUMBER, but will never be changed. Used as a reference to the original file/stream, if any processor has changed the actual file sequence number. Derivation: Mandatory, defaulted by SEQUENCE_NUMBER. Set by the first processor and left unchanged.
TOTAL_NUMBER_OF_RECORDS	9(9)	The total number of Basic Record in the file, excluding header and trailer. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Condition: Not Present in a Notification File or if no Detail records are present. Maximum Number: 999999999 Derivation: Mandatory. Might be recalculated by any processor.
TAP_TOTAL_NUMBER_OF_RECORDS	Integer	Mandatory. Set by TAP input grammar.
FIRST_START_TIMESTAMP	YYYYMMDD HHMISS	The earliest start of charging timestamp on any Basic Detail Record. It is not necessarily the start of charging timestamp of the first charge record on the file. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Condition: Not present in a Notification File or if no Detail records are present. Format: YYYYMMDD HHMISS (see also 'Time-stamp') local time, and not UTC time, is used to determine the earliest call. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory.

Table 79-46 (Cont.) Trailer Record Fields

Name	Format	Description
FIRST_CHARGING_UTC_TIME_OFFSET	X(5)+/-HHMI	All timestamps are sender (VPLMN) local time. So that the time can be equated to time in the recipient (HPLMN) local time, the sender shall give the difference between local time and UTC time. UTC Time Offset = Local Time minus UTC Time. Can be used to translate the TRANSFER_CUTOFF_TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Example: Washington DC, USA 1000hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 10 - 15 = -0500 Madrid, Spain 1600hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 16 - 15 = +0100 Sydney, Australia 0100hrs 11/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = (01 + 24) - 15 = +1000 **Note:** Where dates are different, 24 is added to the time of the greater date Derivation: Mandatory. Set by the first processor and left unchanged.
LAST_START_TIMESTAMP	YYYYMMDDHH MISS	The latest start of charging timestamp on any Basic Detail Record. It is not necessarily the start of charging timestamp of the last charge record on the file. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Might also be used to validate that all records are earlier then the given transfer cutoff timestamp (see header record). Condition: Not present in a Notification File or if no Detail records are present. Format: YYYYMMDD HHMISS (see also 'Time-stamp') local time, and not UTC time, is used to determine the earliest call. Optional Field Usage: It is possible to read/write dates in number of seconds since 01.01.1970 00:00:00; for example, 12345. The internal representation is the format YYYYMMDDHHMISS anyway. This is just an optional input/output format conversion. Derivation: Mandatory.

Table 79-46 (Cont.) Trailer Record Fields

Name	Format	Description
LAST_CHARGING_UTC_TIME_OFFSET	X(5)+/-HHMI	All timestamps are sender (VPLMN) local time. So that the time can be equated to time in the recipient (HPLMN) local time, the sender gives the difference between local time and UTC time. UTC Time Offset = Local Time minus UTC Time. Can be used to translate the TRANSFER_CUTOFF_TIMESTAMP into a unified UTC time. This might be useful if a centralized rating and billing will take place. Example: Washington DC, USA 1000hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 10 - 15 = -0500 Madrid, Spain 1600hrs 10/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = 16 - 15 = +0100 Sydney, Australia 0100hrs 11/10/97 UTC Time 1500hrs 10/10/97 UTC Time Offset = (01 + 24) - 15 = +1000 **Note:** Where dates are different, 24 is added to the time of the greater date. Derivation: Mandatory. Set by the first processor and left unchanged.
TOTAL_RETAIL_CHARGED_VALUE	9(15)	The sum of the Retail Charged Amount Value of any Basic Detail Record. The toll element of a charge is that portion related to the carrier and Destination. There will only be one toll charge present for all chained records. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Values: Space: No price given, like NULL in a database. Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -99999999999999 Maximum: 999999999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory.

Table 79-46 (Cont.) Trailer Record Fields

Name	Format	Description
TOTAL_WHOLESALE_CHARGED_VALUE	9(15)	The sum of the Wholesale Charged Amount Value of any Basic Detail Record contained in the batch. This is present for audit purposes only. Should be used as a check value, to determine that all records have been correctly transmitted and/or used. Values: Variable floating point format: Given value, might be 0.000. The floating decimal point must be set. Minimum: -99999999999999 Maximum: 999999999999999 Examples: '00000000125' for 125,00 '00000012.50' for 12,50 '-0012.56780' for -12,5678 Derivation: Mandatory.
TAP_TOTAL_CHARGE_VALUE	Decimal	Mandatory. Set by TAP input grammar.
TOTAL_TAX_VALUE	Decimal	Calculated. Auto-generated.
TAP_TOTAL_TAX_VALUE	Decimal	Mandatory. Set by TAP input grammar.
TAP_TOTAL_DISCOUNT_VALUE	Decimal	Mandatory. Set by TAP input grammar.
OPERATOR_SPECIFIC_INFO	String	Stores a key to identify the CDR used to generate a specific EDR. Useful for RAP or CIBER return. Optional. Default = ' ' Must be set by an iScript.
CIBER_FILLER	String	Optional.
CREATION_TIMESTAMP	Date	Optional. See the CIBER specification for usage.
CIBER_RECORD_TYPE	String	Optional. See the CIBER specification for usage.
SETTLEMENT_PERIOD	String	Optional. See the CIBER specification for usage.
CLEARINGHOUSE_ID	String	Optional. See the CIBER specification for usage.
CURRENCY	String	Optional. See the CIBER specification for usage.
SENDING_CLEARINGHOUSE_BID	String	Optional. See the CIBER specification for usage.
CIBER_R70 BATCH_TOTALS_SIGN	String	Optional. See the CIBER specification for usage.
CIBER_R70 ORIGINAL_TOTALS_SIGN	String	Optional. See the CIBER specification for usage.
ORIGINAL_SEQUENCE_NUMBER	Integer	Optional. See the CIBER specification for usage.
ORIGINAL_CREATION_TIMESTAMP	Date	Optional. See the CIBER specification for usage.
ORIGINAL_TOTAL_NUMBER_OF_RECORDS	Integer	Optional. See the CIBER specification for usage.
ORIGINAL_TOTAL_WHOLESALE_CHARGED_VALUE	Decimal	Optional. See the CIBER specification for usage.
NOTIFICATION_END_INDEX	Integer	Notification block end index.
AUDIT_CONTROL_INFO_START_INDEX	Integer	AuditControlInfo block start index.

Table 79-46 (Cont.) Trailer Record Fields

Name	Format	Description
AUDIT_CONTROL_INFO_END_INDEX	Integer	AuditControlInfo block end index.
DELAYED_ERROR_BLOCK	String	Stores the block name that has the fatal error.
TOTAL_CHARGE_VALUE_LIST	Block	*n* times. Mandatory. The TAP record is used by GSM operators and data clearinghouses to exchange roaming information.
TOTAL_CHARGE_VALUE	Decimal	Mandatory. Set by TAP grammar.
CHARGE_TYPE	String	Mandatory. Set by TAP grammar.
TOTAL_CHARGE_REFUND	String	Optional.
TOTAL_CHARGE_REFUND	Decimal	Optional.

Associated UTCOffset Record

Table 79-47 lists the fields in the Associated UTCOffset Record.

Table 79-47 Associated UTCOffset Record Fields

Name	Format	Description
UTCTIMEOFFSETCODE	Integer	Stores the UTC time offset code from TAP header.
UTCTIMEOFFSET	String	Stores the UTC time offset value from TAP header.

Associated Recentity Record

Table 79-48 lists the fields in the Associated Recentity Record.

Table 79-48 Associated Recentity Record Fields

Name	Format	Description
RECENTITYCODE	Integer	Stores the REC entity code from TAP header.
RECENTITYTYPE	Integer	Stores the REC entity type from TAP header.
RECENTITYID	String	Stores the REC entity ID from TAP header.

TAP Total Charge Value List

Mandatory. 0 .. *N* times.

Table 79-49 lists the TAP Total Charge Value List fields.

Table 79-49 TAP Total Charge Value List Fields

Name	Format	Description
TOTAL_CHARGE_VALUE	Decimal	Set by TAP grammar. Mandatory.

Table 79-49 (Cont.) TAP Total Charge Value List Fields

Name	Format	Description
CHARGE_TYPE	String	Set by TAP grammar. Mandatory.
TOTAL_CHARGE_REFUND	Decimal	Optional.

Internal Service Control Container

This record is used internally by the framework.

Table 79-50 lists the fields in the Internal Service Control Container.

Table 79-50 Internal Service Controller Container Fields

Name	Format	Description
STREAM_NAME	String	Calculated.
OFFSET_GENERATION	Integer	Calculated.
SEQ_CHECK	Integer	Calculated.
SEQ_GENERATION	Integer	Calculated.
TRANSACTION_ID	Decimal	Calculated.
PROCESS_STATUS	Integer	Values: 0: Normal (default) 1: Recycling 2: Recycling test Mandatory. Calculated.

Customer Data Record

This record is used internally to save all customer-related attributes along with an event.

Table 79-51 lists the fields in the Customer Data Record.

Table 79-51 Customer Data Record Fields

Name	Format	Description
ACCOUNT_ID	String	Mandatory.
ACCOUNT_PARENT_ID	String	Optional.
ACCOUNT_NO	String	Mandatory.
CREATION_DATE	Date	Mandatory.
CURRENCY	String	Mandatory.
CUST_SEG_LIST	String	Mandatory.
RESIDENCE_TYPE	String	Optional.
SYSTEM_BRAND	String	Optional.

Table 79-51 (Cont.) Customer Data Record Fields

Name	Format	Description
BILL_CYCLE	String	Values: 00-28 Mandatory.
BILL_FREQUENCY	Integer	Values: 1-12 Mandatory.
PAYMENT_TYPE	String	Optional.
BILL_STATE	Integer	Mandatory.
ACTG_LAST_DATE	Date	Mandatory.
ACTG_NEXT_DATE	Date	Mandatory.
ACTG_FUTURE_DATE	Date	Mandatory.
ACTG_USED_DATE	Date	Mandatory.
BILL_LAST_DATE	Date	Mandatory.
BILL_NEXT_DATE	Date	Mandatory.
BILL_FUTURE_DATE	Date	Mandatory.
RESOURCE_LIST	String	Optional.
LEAST_COST	Integer	Optional.
PROMOTIONAL_SAVING	Integer	Optional.
PROMOTION	Integer	Optional.

Purchased Charge Offers

Table 79-52 lists the fields in the Purchase Products block.

Table 79-52 Purchased Products Fields

Name	Format	Description
PRODUCT_NAME	String	Mandatory.
USAGE_START	Date	Optional.
USAGE_END	Date	Optional.
QUANTITY	Decimal	Optional.
OFFERING_POID	String	Optional.
OVERRIDDEN_OFFERING_POID	String	Optional.
NODE_LOCATION	String	Mandatory.
DEAL_NAME	String	Mandatory. **Important:** The DEAL_NAME value is not stored in memory nor retained in the EDR. Therefore, this value will not appear in the EDR dump.
PLAN_NAME	String	Mandatory.
PRODUCT_TYPE	Integer	Defines system or normal product; for example, 603 or 602.

Table 79-52 (Cont.) Purchased Products Fields

Name	Format	Description
RATEPLAN_NAME	String	Mandatory.
PRIORITY	Integer	Mandatory.
PRODUCT_ID	String	Optional.
SERVICE_TYPE	String	For example, **/service/telco/gsm/data**. Mandatory.
SERVICE_ID	String	Optional.
SERVICE_PROMO_CODE	String	Optional.
SERVICE_VENDOR	String	Optional.
SERVICE_SOURCE	String	Optional.
SERVICE_LOGIN	String	Mandatory.
SERVICE_MSISDN	String	Optional.
SERVICE_IMSI	String	Optional.
SERVICE_STATUS	String	Mandatory.
SERVICE_USED_ITEM_POID	String	Mandatory.
NETWORK_IDENT	String	Optional.
FIRST_USAGE_INDICATOR	Integer	Specifies whether the charge offer is configured to start when first used and the first-usage validity period status. Optional.

Extended Rating Attributes List

Table 79-53 lists the fields in the Extended Rating Attributes List.

Table 79-53 Extended Rating Attributes List Fields

Name	Format	Description
PROFILE	String	Mandatory.
LABEL	String	Optional.

Profile Attributes

Table 79-54 lists the Profile Attributes fields.

Table 79-54 Profile Attributes Fields

Name	Format	Description
KEY	String	Mandatory.
VALUE	String	Mandatory.

Alias List

Table 79-55 lists the Alias list field.

Table 79-55 Alias List Field

Name	Format	Description
ALIAS_NAME	String	Mandatory.

Discount List

Table 79-56 lists the Discount List fields.

Table 79-56 Discount List Fields

Name	Format	Description
BALANCE_GROUP_ID	String	Mandatory.
DISCOUNT_OWNER_ACCT_ID	String	Mandatory.
DISCOUNT_OWNER_ID	String	Mandatory.
DISCOUNT_OWNER_TYPE	String	Mandatory.

Purchased Discounts

Table 79-57 lists the Purchased Discounts fields.

Table 79-57 Purchased Discounts Fields

Name	Format	Description
DISCOUNT_ID	String	Mandatory.
DISCOUNT_MODEL	String	Mandatory.
PURCHASE_START	Date	Mandatory.
PURCHASE_END	Date	Mandatory.
USAGE_START	Date	Optional.
USAGE_END	Date	Optional.
PRIORITY	Integer	Mandatory.
MODE	Integer	Mandatory.
VALID_FLAG	Integer	Mandatory.
TYPE	Integer	Mandatory.
OFFERING_POID	String	Mandatory.
NODE_LOCATION	String	Mandatory.
STATUS	Integer	Mandatory.
QUANTITY	Integer	Mandatory.
FLAGS	Integer	Mandatory.

Table 79-57 (Cont.) Purchased Discounts Fields

Name	Format	Description
SCALE	Decimal	Mandatory.
FIRST_USAGE_INDICATOR	Integer	Specifies whether the charge offer is configured to start when first used and the first-usage validity period status. Optional.

Sponsor List

Table 79-58 shows the Sponsor List fields.

Table 79-58 Sponsor List Fields

Name	Format	Description
BALANCE_GROUP_ID	String	Mandatory.
SPONSOR_OWNER_ACCT_ID	String	Mandatory.
SPONSOR_OWNER_ID	String	Mandatory.
SPONSOR_OWNER_TYPE	String	Mandatory.

Sponsorship Details

Table 79-59 lists the Sponsorship Details fields.

Table 79-59 Sponsorship Details Fields

Name	Format	Description
SPONSORSHIP_ID	String	Mandatory.
DISCOUNT_MODEL	String	Mandatory.
VALID_FLAG	Integer	Mandatory.

Plan List

Table 79-60 lists the Plan List field.

Table 79-60 Plan List Field

Name	Format	Description
PLAN_ID	String	Mandatory.

Balance Group

Table 79-61 lists the Balance Group field.

Table 79-61 Balance Group Field

Name	Format	Description
BALANCE_GROUP_ID	String	Mandatory.

Balance Element

Table 79-62 lists the Balance Element fields.

Table 79-62 Balance Element Fields

Name	Format	Description
RESOURCE_ID	Integer	Mandatory.
CURR_BAL	Decimal	Mandatory.
CREDIT_FLOOR	Decimal	Mandatory.
CREDIT_LIMIT	Decimal	Mandatory.
RESERVED_AMOUNT	Decimal	Mandatory.
Name	Format	Description

Associated CIBER Extension Record

See the CIBER 2.5 specification for explanations of the fields listed in Table 79-63.

Table 79-63 Associated CIBER Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Default = 701 (this is not yet the final value). Mandatory.
RECORD_NUMBER	Integer	Auto-generated. Mandatory.
FILLER	String	The filler for CIBER optional fields at the end of a record. Optional.
NO_OCC	Integer	Flag to suppress OCC (type 50 or 52) record-creation process. Optional.
INTERN_MOBILE_ID_NO	String	None
INTERN_CALLED_NO	String	None
INTERN_MSISDN_MDN	String	None
INTERN_CALLER_ID	String	None
INTERN_ROUTING_NO	String	None
INTERN_TLDN_NO	String	None
CIBER_RECORD_TYPE	String	Optional.
RETURN_CODE	String	Optional.

Table 79-63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
RETURN_REASON_CODE	String	Optional.
INVALID_FIELD_ID	String	Optional.
HOME_CARRIER_SID	String	Optional.
MOBILE_ID_NO_LENGTH	Integer	Optional.
MOBILE_ID_NO	String	Optional.
MOBILE_ID_NO_OVERFLOW	String	Optional.
ELECTRONIC_SERIAL_NO	String	Optional.
CALL_DATE	Date	Optional.
SERVING_CARRIER_SID	String	Optional.
TOTAL_CHARGE_AND_TAX	Decimal	Optional.
TOTAL_STATE_TAX	Decimal	Optional.
TOTAL_LOCAL_TAX	Decimal	Optional.
CALL_DIRECTION	String	Optional.
CALL_COMPLETION_INDICATOR	String	Optional.
CALL_TERMINATION_INDICATOR	String	Optional.
CALLED_NO_LENGTH	Integer	Optional.
CALLED_NO	String	Optional.
CALLED_NO_OVERFLOW	String	Optional.
TEMP_LOCAL_DIRECTORY_NO	String	Optional.
CURRENCY_TYPE	String	Optional.
ORIG_BATCH_SEQ_NO	Integer	Optional.
INITIAL_CELL_SITE	String	Optional.
TIME_ZONE_INDICATOR	String	Optional.
DAYLIGHT_SAVINGS_INDICATOR	String	Optional.
MSG_ACCOUNTING_DIGITS	String	Optional.
SSU_CONNECT_TIME	Date	Optional.
SERVING_STATE	String	Optional.
RECV_CARRIER_SID	String	Optional.
TRANS_CODE1	String	Optional.
TRANS_CODE2	String	Optional.
SENDING_CARRIER_SID	String	Optional.
CHARGE_NO_1_IND	String	Optional.
CHARGE_NO_1_CONNECT_TIME	Date	Optional.
CHARGE_NO_1_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_1_ELAPSED_TIME	String	Optional.
CHARGE_NO_1_RATE_PERIOD	String	Optional.
CHARGE_NO_1_MULTIRATE_PERIOD	String	Optional.

Table 79-63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
CHARGE_NO_1	Decimal	Optional.
CHARGE_NO_2_IND	String	Optional.
CHARGE_NO_2_CONNECT_TIME	Date	Optional.
CHARGE_NO_2_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_2_ELAPSED_TIME	String	Optional.
CHARGE_NO_2_RATE_PERIOD	String	Optional.
CHARGE_NO_2_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_2	Decimal	Optional.
CHARGE_NO_3_IND	String	Optional.
CHARGE_NO_3_CONNECT_TIME	Date	Optional.
CHARGE_NO_3_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_3_ELAPSED_TIME	String	Optional.
CHARGE_NO_3_RATE_PERIOD	String	Optional.
CHARGE_NO_3_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_3	Decimal	Optional.
CHARGE_NO_4_IND	String	Optional.
CHARGE_NO_4_CONNECT_TIME	Date	Optional.
CHARGE_NO_4_CHARGEABLE_TIME	String	Optional.
CHARGE_NO_4_ELAPSED_TIME	String	Optional.
CHARGE_NO_4_RATE_PERIOD	String	Optional.
CHARGE_NO_4_MULTIRATE_PERIOD	String	Optional.
CHARGE_NO_4	Decimal	Optional.
CHARGE_NO_1_SURCHARGE_IND	String	Optional.
CHARGE_NO_2_SURCHARGE_IND	String	Optional.
CHARGE_NO_3_SURCHARGE_IND	String	Optional.
CHARGE_NO_4_SURCHARGE_IND	String	Optional.
TOLL_CONNECT_TIME	Date	Optional.
TOLL_CHARGEABLE_TIME	String	Optional.
TOLL_ELAPSED_TIME	String	Optional.
TOLL_TARIFF_DESC	String	Optional.
TOLL_RATE_PERIOD	String	Optional.
TOLL_MULTIRATE_PERIOD	String	Optional.
TOLL_RATE_CLASS	String	Optional.
TOLL_FROM_RATING_NPA_NXX	String	Optional.
TOLL_CHARGE	Decimal	Optional.
TOLL_STATE_TAX	Decimal	Optional.
TOLL_LOCAL_TAX	Decimal	Optional.

Table 79-63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
TOLL_NETWORK_CARRIER_ID	String	Optional.
MSID_INDICATOR	String	Optional.
MSID	String	Optional.
MSISDN_MDN_LENGTH	Integer	Optional.
MSISDN_MDN	String	Optional.
ESN_IMEI_INDICATOR	String	Optional.
ESN_IMEI	String	Optional.
CALLER_ID_LENGTH	Integer	Optional.
CALLER_ID	String	Optional.
ROUTING_NO_LENGTH	Integer	Optional.
ROUTING_NO	String	Optional.
TLDN_NO_LENGTH	Integer	Optional.
TLDN_NO	String	Optional.
AIR_CONNECT_TIME	Date	Optional.
AIR_CHARGEABLE_TIME	String	Optional.
AIR_ELAPSED_TIME	String	Optional.
AIR_RATE_PERIOD	String	Optional.
AIR_MULTIRATE_PERIOD	String	Optional.
AIR_CHARGE	Decimal	Optional.
OTHER_CHARGE_1_INDICATOR	String	Optional.
OTHER_CHARGE_1	Decimal	Optional.
CALLED_COUNTRY	String	Optional.
SERVING_COUNTRY	String	Optional.
TOLL_RATING_POINT_LENGTH	Integer	Optional.
TOLL_RATING_POINT	String	Optional.
FEATURE_USED_AFTER_HO_IND	String	Optional.
OCC_START_DATE	Date	Optional.
OCC_CHARGE	Decimal	Optional.
FET_EXEMPT_INDICATOR	String	Optional.
PASS_THROUGH_CHARGE_IND	String	Optional.
CONNECT_TIME	Date	Optional.
RECORD_USE_INDICATOR	String	Optional.
OCC_DESCRIPTION	String	Optional.
OCC_END_DATE	Date	Optional.
RECORD_CREATE_DATE	Date	Optional.
SEQ_INDICATOR	String	Optional.
OCC_INTERVAL_INDICATOR	String	Optional.

Table 79-63 (Cont.) Associated CIBER Extension Record Fields

Name	Format	Description
EVENT_DATE	Date	Optional.
MIN_ESN_APP_INDICATOR	String	Optional.
R70_RECORD_USE_INDICATOR	String	Optional.
EVENT_TIME	Date	Optional.

Discount Balance Packet

Table 79-64 lists the Discount Balance Packet fields.

Table 79-64 Discount Balance Packet Fields

Name	Format	Description
DISCOUNT_KEY	String	Discount key (normally the account ID).
ACCOUNT_ID	Integer	Related account ID.
RESOURCE_ID	Integer	BRM mapped balance element ID.
DISCOUNT_STEP	Integer	Alternative key if balance element ID 0.
DISCOUNT_MASTER	Integer	Alternative key if balance element ID 0.
UPDATE_LEVEL	String	Always empty. Supports backward compatibility.
SERVICE_ID	Integer	Supports backward compatibility.

Aggregation Period

Table 79-65 lists the Aggregation Period fields.

Table 79-65 Aggregation Period Fields

Name	Format	Description
PERIOD	String	The accounting cycle. *YYYYMMDD*.
ITEM_POID	String	POID of the item that identifies the accounting cycle.
CREATED	String	Creation date.
TOTAL_CHARGE	Decimal	Total charge.
TOTAL_QUANTITY	Decimal	Total quantity based on RUMs in the discount filter.
TOTAL_EVENT	Decimal	Sum of events based on charge packets.
GRANTED_CHARGE	Decimal	The discounted charge.
GRANTED_QUANTITY	Decimal	The discounted quantity.
FRAME_CHARGE	Decimal	Total charge of the discount frame, based on the frame.
FRAME_QUANTITY	Decimal	Total quantity in the discount frame based on RUMs in the discount filter.
FRAME_EVENT	Decimal	Total events of the discount frame, based on charge packets.

Discount Packet

Table 79-66 lists the Discount Packet fields.

Table 79-66 Discount Packet Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory.
CREATED	String	Creation date.
OBJECT_ID	String	Discount/sponsor object ID.
OBJECT_TYPE	String	Discount/sponsor object that generated the event.
OBJECT_ACCOUNT	Integer	POID of the discount owner.
OBJECT_OWNER_ID	Integer	POID type of the discount owner.
OBJECT_OWNER_TYPE	String	POID type of the discount owner.
DISCOUNTMODEL	String	Discount.
DISCOUNTRULE	String	Discount rule.
DISCOUNTSTEPID	Integer	Discount step ID.
DISCOUNTBALIMPACTID	Integer	Discount balance impact ID.
TAX_CODE	String	Tax code.
GRANTED_AMOUNT	Decimal	Granted discount/sponsorship amount. Can be currency or noncurrency.
GRANTED_AMOUNT_ORIG	Decimal	Original granted discount/sponsorship amount. Used when exchange rate is configured.
GRANTED_QUANTITY	Decimal	The discount base value used to compute the granted amount.
AMOUNT	Decimal	Discounted currency amount.
PIN_PERCENT	Decimal	Percent value filled from charge packet
QUANTITY	Decimal	Discounted noncurrency amount.
QUANTITY_FROM	Decimal	Discounted quantity start value.
QUANTITY_TO	Decimal	Discounted quantity end value.
VALID_FROM	Date	Grant case, valid-from date.
VALID_TO	Date	Grant case, valid-to date.
VALID_FROM_DETAIL	Integer	First Usage Offset and Unit for valid-from date.
VALID_TO_DETAIL	Integer	First Usage Offset and Unit for valid-to date.
CYCLE_OFFSET	Integer	Charge offer cycle that identifies a grant's validity period.
BALANCE_GROUP_ID	Integer	Balance group ID.
SERVICE_CODE	String	Service code.
RESOURCE_ID	Integer	Balance Element ID.
RESOURCE_ID_ORIG	Integer	Original balance element ID. Used when exchange rate is configured.
ZONEMODEL_CODE	String	Zone model.
IMPACT_CATEGORY	String	Impact category.
TIMEZONE_CODE	String	Time-zone code.

Table 79-66 (Cont.) Discount Packet Fields

Name	Format	Description
TIMEMODEL_CODE	String	Time model code.
SERVICE_CLASS	String	Service class.
PRICEMODEL_CODE	String	Pricing code.
RUM	String	RUM.
RATETAG	String	Rate tag.
RATEPLAN	String	Charge.
GLID	String	G/L ID.
OFFERING_POID	String	Purchased discount offer POID.
NODE_LOCATION	String	Node location.
INTERN_PACKET_INDEX	Integer	Packet ID.
INTERN_SRC_PACKET_INDEX	Integer	Source packet ID.
INTERN_RUM_ID	Integer	RUM ID passed in and out, used in real-time pipeline only.
INTERN_DISC_MATCH_FACTOR	Decimal	Discount match factor (the percentage of usage discounted).
INTERN_TOTAL_MATCH_FACTOR	Decimal	Total discounted match factor (the total percentage of usage discounted).
DEFERRED_AMOUNT	Decimal	Deferred amount.

Discount Sub-Balance Packet

Table 79-67 lists the Discount Sub-Balance Packet fields.

Table 79-67 Discount Sub-Balance Packet Fields

Name	Format	Description
REC_ID	Integer	Record ID.
VALID_FROM	Date	Validity start time.
VALID_TO	Date	Validity end time.
AMOUNT	Decimal	Amount.
CONTRIBUTOR	String	Contributor.
NEXT_BAL	Date	Next balance.
DELAYED_BAL	Decimal	Delayed balance.
GRANTOR	String	The charge offer or discount offer that granted this balance element.
VALID_FROM_DETAILS	Date	Sub-balance start time mode (such as first-usage or relative) and relative offset and unit.
VALID_TO_DETAILS	Date	Sub-balance end time mode (such as relative) and relative offset and unit.
GRANT_VALID_FROM	Date	Grant validity start time.
GRANT_VALID_TO	Date	Grant validity end time.

Associated SMS Extension Record

Table 79-68 lists the Associated SMS Extension Record fields.

Table 79-68 Associated SMS Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 580.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
CONTENT_INDICATOR	String	Optional.
ORIGINATING_SWITCH_IDENTIFICATION	String	Optional.
DESTINATION_SWITCH_IDENTIFICATIO N	String	Optional.
PROVIDER_ID	String	Optional.
SERVICE_ID	String	Optional.
DEVICE_NUMBER	String	Optional.
PORT_NUMBER	String	Optional.
DIALED_DIGITS	String	Optional.

Associated MMS Extension Record

Table 79-69 lists the Associated MMS Extension Record fields.

Table 79-69 Associated MMS Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Default = 590.
RECORD_NUMBER	Integer	Mandatory. Auto-generated.
ACCOUNT_STATUS_TYPE	String	Optional.
PRIORITY	String	Optional.
MESSAGE_CONTENT	String	Optional.
MESSAGE_ID	String	Optional.
STATION_IDENTIFIER	String	Optional.
FC_INDICATOR	String	Optional.
CORRELATION_ID	String	Optional.
CELL_ID	String	Optional.
B_CELL_ID	String	Optional.
A_TERM_CELL_ID	String	Optional.
DEVICE_NUMBER	String	Optional.

Table 79-69 (Cont.) Associated MMS Extension Record Fields

Name	Format	Description
PORT_NUMBER	String	Optional.

SGSN Information

Table 79-70 lists the SGSN Information field.

Table 79-70 SGSN Information Field

Name	Format	Description
SGSN_ADDRESS	String	Mandatory.

Profile Event Ordering

Table 79-71 lists the Profile Event Ordering fields.

Table 79-71 Profile Event Ordering Fields

Name	Format	Description
RECORD_TYPE	String	Mandatory. Must be set to 850.
RECORD_NUMBER	Integer	Mandatory.
BAL_GRP_POID	String	Mandatory.
CRITERIA_NAME	String	Mandatory.
PROFILE_POID	String	Mandatory.
BILLING_CYCLE_TIMESTAMP	Date.	Mandatory.

Associated Roaming Extension Record

Table 79-72 lists the Associated Roaming Extension Record fields.

Table 79-72 Associated Roaming Extension Record Fields

Name	Format	Description
RECORD_TYPE	String	None
RECORD_NUMBER	Integer	None
TAP_FILE_SEQ_NO	Integer	None
RAP_FILE_SEQ_NO	Integer	None
RAP_RECORD_TYPE	String	None
SENDER	String	None
RECIPIENT	String	None

Table 79-72 (Cont.) Associated Roaming Extension Record Fields

Name	Format	Description
TAP_FILE_PATH	String	None
START_MISSING_SEQ_NUM	Integer	None
END_MISSING_SEQ_NUM	Integer	None
SUSPENSION_TIME	Date	None
PORT_NUMBER	String	None
TOTAL_TAX_REFUND	Decimal	None
TOTAL_DISCOUNT_REFUND	Decimal	None
GUARANTEED_BIT_RATE	String	None
MAXIMUM_BIT_RATE	String	None
HSCSD_INDICATO	String	None
SMS_ORIGINATOR	String	None
SMS_DESTINATION_NUMBER	String	None
DISCOUNTABLE_AMOUNT	Decimal	None
DISCOUNT_CODE	Integer	None
NETWORKACCESS_IDENTIFIE	String	None
ISM_SIGNALLING_CONTEXT	Integer	None
IMSI	String	None
HOME_BID	String	None
HOMELOCATION_DESCRIPTION	String	None
MOBILE_ID_NUMBER	String	None
MOBILE_DIR_NUMBER	String	None
TOTAL_ADVISEDCHARGE	Decimal	None
TOTAL_ADVISEDCHARGE_REFUND	Decimal	None
TOTAL_COMMISSION	Decimal	None
TOTAL_COMMISSION_REFUND	Decimal	None
ITEM_OFFSET	Integer	None
ERROR_CODE	Integer	None
TOTAL_SEVERE_RETURN_VALUE	Decimal	None
RETURN_DETAILS_COUNT	Integer	None
CLIR_INDICATOR	String	None
TAP_CURRENCY	String	Currency used for TAP3 and TAP 311.

Associated RAP Extension

Table 79-73 lists the Associated RAP Extension fields.

Table 79-73 Associated RAP Extension Fields

Name	Format	Description
PATH_ITEMID	Integer	None
ITEM_OCCURRENCE	Integer	None
ITEM_LEVEL	Integer	None

Total Advised Charge Value List

Table 79-74 lists the Total Advised Charge Value List fields.

Table 79-74 Total Advised Charge Value List Fields

Name	Format	Description
TOTAL_ADVISEDCHARGE	Decimal	None
TOTAL_ADVISEDCHARGE_REFUND	Decimal	None
ADVISED_CHARGE_CURRENCY	String	Optional. AdvisedChargeCurrency item.
TOTAL_COMMISSION;	Decimal	None
TOTAL_COMMISSION_REFUND	Decimal	None

Field Usage

Roaming

The following conditions are checked to determine if the usage record is for roaming:

```
if [DETAIL.USAGE_DIRECTION] = 2
       then Roaming = TRUE; Roaming-Type = MOC
elseif [DETAIL.USAGE_DIRECTION] = 3
       then Roaming = TRUE; Roaming-Type = MTC
else   Roaming = FALSE
```

International-Call

The following conditions are checked to determine if the usage record is for an international call:

```
if [DETAIL.CONNECT_SUB_TYPE] = '04'
       then International-Call=TRUE
```

CLI Normalization

The result of the mapping operation (performed either within an input module or within an iScript) must be written/copied to the following internal container fields:

DETAIL.A_NUMBER -> normalize -> DETAIL.INTERN_A_NUMBER_ZONE

DETAIL.B_NUMBER -> normalize -> DETAIL.INTERN_B_NUMBER_ZONE

DETAIL.C_NUMBER -> normalize -> DETAIL.INTERN_C_NUMBER_ZONE

The original values within the Basic Detail Record are kept unchanged.

The following fields determine how and which a normalization function should be carried out:

DETAIL.A_NUMBERING_PLAN -> normalize DETAIL.A_NUMBER

DETAIL.B_NUMBERING_PLAN -> normalize DETAIL.B_NUMBER

DETAIL.C_NUMBERING_PLAN -> normalize DETAIL.C_NUMBER

if DETAIL.x_NUMBERING_PLAN between 1 and 9 -> use "ISDN, MSISDN"

if DETAIL.x_NUMBERING_PLAN between A and B -> use "Ipv4, IPv6"

if DETAIL.x_NUMBERING_PLAN = 0 -> no normalization

ISDN, MSISDN

The following rules normalize the A number and B number. All CLIs are normalized to match the international format:
International_access_codeCountry_codeNational_destination_codeSubscriber_number; for example, '00491711234567' or '004980012345'.

To handle a flexible international format, the following parameters must be set prior to the normalization:

International_access_code (iac): international access code; for example, '00'

> **Note:**
>
> Multiple iacs might be defined.

International_access_code_sign (iacs): international access code sign; for example, '+'

Country_code (cc): country code of the home country; for example, '49'

National_destination_access_code (ndac): national destination access code for long distance; for example, '0'

National_destination_code (ndc): default national destination code; for example, '172' (only for special mobile calls)

Normal-Call

Table 79-75 lists the Normal-Call fields.

Table 79-75 Normal-Call Fields

Item	Description
A#:	[DETAIL.A_NUMBER, X(40)]
B#:	[DETAIL.B_NUMBER, X(40)]

1. if <empty>-> replace with '<iac><cc>' (break)

2. if Prefix = '<iacs>'-> replace with '<iac>' (continue)
3. if Prefix = '<iac>'-> do nothing (break)
4. if Prefix = '<ndac>'-> replace with '<iac><cc>' (break)
5. if [TYPE_OF_NUMBER] = 1-> prefixing '<iac>' (break)

Roaming-Call (Mapping only for Zone- and PrefixDesc.-Determination)

An Associated GSM/Wireline Extension Record must exist.

Table 79-76 lists the Roaming-Call fields.

Table 79-76 Roaming-Call Fields

Item	Type	Description
A#:	MOC	-> '0000' + Left([ASS_GSMW_SE.ORIGINATING_SWITCH_ID], 5)
A#:	MTC	-> Normalization as for normal calls
B#:	MOC	-> Normalization as for normal calls
B#:	MTC	-> '0000' + Left([ASS_GSMW_SE.TERMINATING_SWITCH_ID], 5)

If on the input side there is only one single MSC_ID or PLMN_ID available, the related input module has to map this single value into both fields (Originating and Terminating).

Special-Mobile-Call:

A/B-MODIFICATION_INDICATOR = '04'

Normalize to: <iac><cc>'0'<ndc><number>; for example, '0049017222255'

Table 79-77 lists the Special-Mobile-Call fields.

Table 79-77 Special-Mobile-Call Fields

Item	Description
A#:	[DETAIL.A_NUMBER, X(40)]
B#:	[DETAIL.B_NUMBER, X(40)]

1. if <empty>-> replace with '<iac><cc>' (break)
2. if Prefix = '<iacs>'-> replace with '<iac>' (continue)
3. if Prefix = '<iac><cc>'-> replace with '<iac><cc>0' (break)
4. if Prefix = '<ndac>'-> prefixing '<iac><cc>' (break)
5. if Prefix != '<cc>'-> prefixing '<iac><cc>0<ndc>' (break)
6. if [TYPE_OF_NUMBER] = 1-> prefixing '<iac>' (break)

IPv4, IPv6

If the A# (source ip) and B# (destin ip) are carrying ip-addresses, they are normalized to zero-leading-tokens without the dots or colon; for example, '192.168.10.1' is normalized to '192168010001'.

Notations are listed in Table 79-78:

Table 79-78 IPv4 and IPv6 Attributes

Item	Type	Description
IPv4:	nnn.nnn.nnn.nnn	n = (0..9)
IPv6:	hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh:hhhh	h = (0..F)
IPv4 as v6:	0000:0000:0000:0000:0000:0000:nnn.nnn.nnn.nnn	None
or	0000:0000:0000:0000:0000:FFFF:nnn.nnn.nnn.nnn	None

IPv4 Record

Table 79-79 lists the IPv4 Record fields.

Table 79-79 IPv4 Record Fields

Item	Type	Description
A#:	[DETAIL.A_NUMBER, X(40)]	(source ip-address)
B#:	[DETAIL.B_NUMBER, X(40)]	(destination ip-address)

1. determine the four decimal ip-tokens
2. fill each token with leading zeros (up to 3-digits)
3. remove all dots '.'

IPv6 Record

Table 79-80 lists the IPv6 Record fields.

Table 79-80 IPv6 Record Fields

Item	Type	Description
A#:	[DETAIL.A_NUMBER, X(40)]	(source ip-address)
B#:	[DETAIL.B_NUMBER, X(40)]	(destination ip-address)

1. determine the eight hexadecimal ip-tokens
2. fill each token with leading zeros (up to 4-digits)
3. remove all colons ':'

IPv4 as IPv6 Record

Table 79-81 lists the IPv4 vs IPv6 Record fields.

Table 79-81 IPv4 as IPv6 Record Fields

Item	Type	Description
A#:	[DETAIL.A_NUMBER, X(40)]	(source ip-address)
B#:	[DETAIL.B_NUMBER, X(40)]	(destination ip-address)

1. determine the six fix hexadecimal v6 ip-tokens
2. fill each token with leading zeros (up to 4-digits)

3. remove all colons ':'
4. determine the four decimal ip-tokens at the end of the address
5. take the first two v4 tokens and convert them to hexadecimal (for example, '192.168' = 192*256^1 + 168*256^0 = 49320 = 'C0A8')
6. take the last two v4 tokens and convert them to hexadecimal
7. replace the origin v4 address by the two calculated v6 equivalents filled with leading zeros

UsageClass (CallClass)

The result of the mapping operation (performed either within an input module or within an early iScript) must be written/copied to the following internal container fields:

DETAIL.USAGE_CLASS -> mapping -> DETAIL.INTERN_USAGE_CLASS

The original values within the Basic Detail Record are kept unchanged.

The following rules apply to determine the external UsageClass value:

Always: Value of the field [DETAIL.USAGE_CLASS]

ServiceCode / ServiceClass

The result of the mapping operation (performed either within an input module or within an early iScript) must be written/copied to the following internal container fields:

DETAIL.BASIC_SERVICE -> mapping -> DETAIL.INTERN_SERVICE_CODE

mapping -> DETAIL.INTERN_SERVICE_CLASS

The original values within the Basic Detail Record are kept unchanged.

The following rules apply to determine the external ServiceCode value:

Always: Value of the field [DETAIL.BASIC_SERVICE];

containing [SERVICE_TYPE, X(1)]+ [SERVICE_CODE, X(2)]

80

List of Pipeline Manager Modules, iScripts, and iRules

This chapter lists the Oracle Communications Billing and Revenue Management Pipeline Manager modules.

For information about placement of modules in a pipeline, see "Function Module Dependencies".

Pipeline Manager Modules

Table 80-1 lists the Pipeline Manager modules with descriptions.

Table 80-1 Pipeline Manager Modules

Module	Description
Controller	Controls and monitors the pipeline framework.
Database Connect (DBC)	Provides database connections for other modules.
DAT_AccountBatch	Provides customer data from the BRM database.
DAT_AccountRealtime	Provides data to a real-time discounting pipeline. See "Configuring a Real-Time Discounting Pipeline".
DAT_BalanceBatch	Maintains balance information in the Pipeline Manager memory. See "Configuring Discounting Modules and Components".
DAT_BalanceRealtime	Retrieves current balance information from the BRM database and supplies the data to the real-time discounting pipeline. See "Configuring a Real-Time Discounting Pipeline".
DAT_Calendar	Provides special day calendar data for the FCT_MainRating module.
DAT_Currency	Converts currency symbols to numeric values.
DAT_Dayrate	Provides special day rate data for the FCT_Dayrate module.
DAT_Discount	Provides data for the FCT_Discount module and the FCT_DiscountAnalysis module. See "Configuring Discounting Modules and Components".
DAT_ExchangeRate	Provides currency exchange rate data for the FCT_ExchangeRate module.
DAT_InterConnect	Provides network configuration data for the FCT_CarrierIcRating module.
DAT_ItemAssign	Returns the item POID for an item tag to the FCT_ItemAssign and FCT_Billing Record modules.
DAT_Listener	Listens to business events from BRM and provides data to the DAT_AccountBatch and DAT_Discount modules.
DAT_ModelSelector	Provides model selector rules to other modules. See "Configuring Pipeline Rating" and "Configuring Discounting Modules and Components".

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
DAT_NOSP	Provides data for mapping network source and destinations to new values for the FCT_NOSP module. See Identifying the Network Operator/Service Provider .
DAT_NumberPortability	Provides number portability data to the FCT_NumberPortability module. See "Setting Up Number Portability".
DAT_PortalConfig	Provides data for mapping phone number prefixes to descriptions, used by the FCT_PrefixDesc module. See "Creating Call Destination Descriptions".
DAT_PriceModel	Provides price model data for the FCT_MainRating module. See "About Pipeline Rating".
DAT_Rateplan	Provides charge data for the FCT_MainRating module. See "Configuring Pipeline Rating".
DAT_Recycle	Used by standard recycling and Suspense Manager EDR to recycle EDRS. See "Configuring Standard Recycling".
DAT_ScenarioReader	Provides aggregation scenario data for the FCT_AggreGate module. See "Setting Up Pipeline Aggregation".
DAT_TimeModel	Provides time model, time zone, and day code data for the FCT_Mainrating module.
DAT_USC_Map	Provides usage scenario (USC) mapping data. It retrieves USC mapping data from the Pipeline Manager database or an ASCII file for the FCT_USC_Map module.
DAT_Zone	Provides zone data for the FCT_MainRating module.
EDR Factory	Generates and allocates memory to EDR Containers.
Event Handler	Starts external programs.
EXT_InEasyDB	Handles pipeline input from a database. See "Configuring EDR Input Processing". Configure this module as a submodule of the INP_GenericStream module. See INP_GenericStream.
EXT_InFileManager	Performs file handling for pipeline input from files. See "Configuring EDR Input Processing". Configure this module as a submodule of the INP_GenericStream module. See INP_GenericStream.
EXT_OutFileManager	Handles files for the OUT_Generic_Stream and OUT_Reject modules. See "Configuring EDR Output Processing".
Pipeline Dispatcher	Parses CDR files from a single input directory to multiple pipelines.
FCT_Account	Adds customer data to an EDR.
FCT_AccountRouter	For a multischema system, finds the database schema for the customer and routes the EDRs to the appropriate pipeline. See "Using Pipeline Manager with Multiple Database Schemas".
FCT_AggreGate	Performs aggregation of data in EDR containers. See "Setting Up Pipeline Aggregation".

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_APN_Map	**Before zoning**: Maps the access point name (APN) to a physical PDP address. **After zoning**: Enhances zone values to support enhanced zoning functionality.
FCT_ApplyBalance	Reads the discount packets added by DAT_Discount, adds the discounting sub-balance impact to the EDR, and updates the in-memory balance.
FCT_BillingRecord	Consolidates balance impact data into an associated BRM billing record and one or more balance impact packets. This data is loaded into the BRM database by RE Loader. See "About Consolidation for BRM Billing".
FCT_CallAssembling	Assembles EDRs that have been split into multiple EDRs. See "Assembling EDRs".
FCT_CarrierIcRating	Adds roaming data to EDRs for rating by the FCT_PreRating and FCT_MainRating modules.
FCT_CiberOcc	The FCT_CiberOcc module creates a CIBER record for other charges and credits (OCC record), type 50 or 52.
FCT_CliMapping	Maps multiple numbers to a single number for billing. See "Mapping Multiple Phone Numbers to a Single Number".
FCT_CreditLimitCheck	Performs credit limit checking to determine whether the event owner has enough resources for the requested service.
FCT_CustomerRating	Supplies the charge for the FCT_MainRating module. See "About Customer Rating".
FCT_Dayrate	Calculates charges for special day rates, for example, a discount for calls made on January 1.
FCT_Discard	Discards or skips EDRs based on configurable EDR properties. • Skipping an EDR removes it from the pipeline. • Discarding an EDR sends it to a different output stream. In both the cases the state of the EDR becomes invalid. See "Discarding and Skipping EDRs ".
FCT_Discount	Performs discounting functions. See "Configuring Discounting Modules and Components".
FCT_DiscountAnalysis	Performs discounting analysis functions. See "Configuring Discounting Modules and Components".
FCT_DroppedCall	Identifies dropped calls and continuation calls.
FCT_DuplicateCheck	Checks for duplicate EDRs. See "Handling Duplicate EDRs".
FCT_EnhancedSplitting	Specifies different output streams for EDRs based on rules. For example: • You can split EDRs for different service types to different output streams. • You can split EDRs from roaming outcollects and incollects into different streams. See "Using Rules to Send EDRs to Different Output Streams".
FCT_ExchangeRate	Converts the currency used for rating to the home (system) currency, and the customer's billing currency.

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_Filter_Set	Determines whether an EDR is eligible for the system charge offers and system discounts contained in a filter set, and if it is, adds those system charge offers and discounts to a customer's list of purchased charge offers. See "About Using Filter Sets to Apply System Products and Discounts".
FCT_GlobalRating	Rates all EDRs with a default set of charges. See "About Global Rating".
FCT_IRules	Evaluates iRules. Those rules can be used for mapping functions for EDR data fields, splitting EDR containers to different output streams, and so forth.
FCT_IScript	Runs iScripts. The scripts are run in the order specified in the registry.
FCT_Reject	Retrieves an item POID for an item tag from the DAT_ItemAssign module and populates the EDR container with the item POID. See DAT_ItemAssign.
FCT_MainRating	Performs the main Pipeline Manager rating functionality. See "About Main Rating".
FCT_MainZoning	Performs zoning for multi-segment zoning.
FCT_NOSP	Maps network source and destination to new values. See "Identifying the Network Operator/Service Provider ".
FCT_NumberPortability	Specifies the new network operator for an existing phone number. See "Setting Up Number Portability".
FCT_PrefixDesc	Maps phone number prefixes to destination descriptions. See "Creating Call Destination Descriptions".
FCT_PreRating	Calculates zones and creates impact categories.
FCT_PreRecycle	Used for pipeline-only implementations. Gets the file of rejected EDRs from the reject stream output directory. The module puts the reject EDR file into the pipeline input directory for recycling. It uses the same input folder as the incoming CDR files. See "Recycling EDRs in Pipeline-Only Systems".
FCT_PreSuspense	When used as part of BRM standard recycling, this module adds suspense-related information to EDRs. When used with Suspense Manager, this module also configures the queryable fields for EDRs suspended in a specific pipeline.
FCT_RateAdjust	Adjusts the charge for an EDR after rating has been performed. See "About Rate Adjustment" in *BRM Setting Up Pricing and Rating*.
FCT_Recycle	Used for pipeline-only implementations. Runs at the end of the pipeline It does either of the following: • When the FCT_PreRecycle module runs in test mode, the FCT_Recycle module creates a report about the processing, but does not send the EDRs to an output file. • When the FCT_PreRecycle module runs in recycle mode, the FCT_Recycle module sends the results to an output file, and attaches a sequence number to the output file. See "Recycling EDRs in Pipeline-Only Systems".

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
FCT_Reject	The FCT_Reject module analyzes the errors in an EDR and, if necessary, moves the EDR to a reject file. See "About Standard Recycling".
FCT_Rounding	Performs rounding for rating and discounting.
FCT_RSC_Map	Performs rate service class (RSC) mapping. See "About Rate-Service Class Mapping".
FCT_SegZoneNoCust	Finds the segment using the source network information instead of using the customer information.
FCT_ServiceCodeMap	Maps external service codes to internal service codes.
FCT_SocialNo	Flags social numbers for special processing. See "Setting Up Social Numbers".
FCT_Suspense	When used as part of BRM standard recycling, routes failed EDRs to appropriate output streams depending on their processing status (normal, recycling, or test recycling) and suspense status (succeeded or suspended). When used with Suspense Manager, also adds the suspense reason and subreason codes to EDRs.
FCT_TriggerBill	Sends EDRs to the billing-trigger output stream to trigger immediate billing for the associated accounts. It also sets a billing-trigger error code used to route the EDRs to the suspense output stream, and the **Trigger_Billing** recycle key used to retrieve the suspended EDRs for recycling.
FCT_UoM_Map	Converts the unit of measurement (UoM) of an incoming EDR to a UoM needed for rating a particular service.
FCT_UsageClassMap	The FCT_UsageClassMap module maps external codes for secondary services, such as call forwarding, to internal usage classes.
FCT_USC_Map	The FCT_USC_Map module performs usage scenario mapping.
FCT_Zone	The FCT_Zone module computes zones when you use Pipeline Manager only for zoning.
INP_GenericStream	Provides the input interface to the pipeline. See "Configuring EDR Input Processing".
INP_Realtime	Converts data in an flist to the EDR container format. See "Configuring a Real-Time Discounting Pipeline".
INP_Recycle	Used by standard recycling and Suspense Manager in the pre-recycling pipeline. It reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.
IRL_EventTypeSplitting	Sends EDRs to separate output streams based on service codes. See "Sending EDRs to Pipeline Output Streams".
IRL_LeastCostPerEDR	Flags all EDRs that satisfy the criteria for least cost rating. see "About Least Cost Rating".
IRL_PipelineSplitting	Used in the pre-recycling pipeline to send EDRs to different output streams depending on their original pipeline names. The EDRs are then routed to their original pipelines for recycling.
IRL_LeastCostPerEDR	Flags all EDRs that satisfy the criteria for a promotional savings calculation. See "About Least Cost Rating".

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
IRL_UsageType	Assigns usage types to EDRs.
ISC_AddCBD	Prepares EDRs for rerating in the back-out pipeline. **Important:** This is a deprecated module but remains in BRM for backward compatibility.
ISC_BACKOUTTypeSplitting	Used by the backout pipeline for back-out-only rerating. It determines if the EDRs are flagged for back-out-only rerating and sends the EDRs to different output streams based on the event types.
ISC_CiberInputValidation	Performs record-level validations of CIBER records.
ISC_CiberOutputMapping	Adds charge data to the ASSOCIATED_CIBER_EXTENSION block of the EDR. If the EDR does not contain an ASSOCIATED_CIBER_EXTENSION block, this iScript adds one.
ISC_CiberRejectReason	Sets a reason code in the CIBER extension block for records that are rejected.
ISC_EDRToTAPOUTMap	Populates standard values to fields in output TAP file based on its corresponding value in the EDR container.
ISC_GetCamelFlag	Retrieves the CAMEL flag information for a roaming partner. This iScript is used by roaming outcollect processing.
ISC_LeastCost	Performs one of the following: • Calculates and finds the lowest charge for an EDR. • Calculates the total savings when using an overlay promotion. See "About Least Cost Rating" and "About Calculating the Promotional Savings".
ISC_MapNetworkOperatorInfo	Maps the DETAIL.SOURCE_NETWORK field to the PIN_FLD_ORIGIN_NETWORK field and the DETAIL.DESTINATION_NETWORK field to the PIN_FLD_DESTINATION_NETWORK field of the opcode input block for the corresponding event. See "Setting Up Number Portability".
ISC_NRTRDE_ErrorReport	Collects the validation errors in the EDRs and creates error records in the Pipeline Manager database. This iScript is used during roaming incollect processing by the NRTRDE (Near Real-Time Roaming Data Exchange) processing pipeline.
ISC_NRTRDE_EventSplit	Duplicates and routes EDRs to the corresponding roaming partner NRTRDE output streams based on the roaming partner's NRTRDE flag. This iScript is used by roaming outcollect processing.
ISC_NrtrdeHeaderValidation_v2_01	Validates the information in the header record of the TD35 file based on the TD35 specifications. This iScript is used during roaming incollect processing by the NRTRDE processing pipeline.
ISC_ObjectCacheTypeOutputSplitter	Creates two output CDRs from a single input EDR.
ISC_OverrideRateTag	Populates the RATE_TAG field with the value of the NRTRDE flag in the balance impact. This iScript is used by the outcollect settlement pipelines.
ISC_ProfileAnalyzer	Analyzes friends and family extended rating attributes (ERAs) during pipeline rating.
ISC_ProfileLabel	Analyzes ERAs during pipeline rating to determine whether the ERA profiles specified in the **ProfileName** registry entry match the EDR field value.

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
ISC_PostRating	Adds all the retail and wholesale charges and puts them in DETAIL.RETAIL_CHARGED_AMOUNT_VALUE and DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE fields. See "Billing Consolidation with CIBER Roaming and Revenue Assurance".
ISC_SetAndValidateBatchInfo	Populates and validates the batch related fields for the EDR container.
ISC_SetEDRStatus	Sets the EDR status to **Success**, **Suspense**, **Duplicate**, **Discard**, or **Skipped** for each EDR.
ISC_SetOutputStream	Sets the Output Stream to **TelOut**, **SMSOut**, **GPRSOut**, **RejectOut**, or **DuplicateOut** for each EDR.
ISC_SetRevenueFigures	Collects the previous and current charged and discount amount for a configured Balance Element ID.
ISC_SetRevenueStream	Sets the Revenue Stream to **Retail**, **Wholesale**, **Roaming**, or **Unknown** for each EDR.
ISC_SetSvcCodeRTZoning	Finds the service type and updates the **DETAIL.INTERN_SERVICE_CODE** EDR field with the customized service code value for each EDR.
ISC_TapDetailValidation_v3_12	Validates that the fields present in the detail record of the EDR container contain valid data.
ISC_TapSplitting	Splits mobile originating and terminating EDRs when the CDR contains more than one basic service. ISC_TapSplitting creates a new EDR for each additional basic service. See "Generating Multiple TAP MOC and MTC Records".
ISC_TaxCalc	Applies a flat tax to pipeline-rated events.
LOG	Logs error messages.
Memory Monitor	Monitors Pipeline Manager system memory during startup and while it is processing files.
NET_EM	The NET_EM module hosts a BRM External Module (EM). This allows the NET_EM module to use the BRM API opcodes to transfer data between real-time rating and Pipeline Manager. See "Configuring a Real-Time Discounting Pipeline".
OUT_DB	Sends output to the database. See "Sending Output to a Database".
OUT_DevNull	Removes EDRs that are not needed by Pipeline Manager. See "Configuring Output of Discarded EDRs". All registry entries and error messages are handled by the Output Collection module. See Output Collection. For more information, see "Discarding and Skipping EDRs ".
OUT_GenericStream	Handles the output stream for rated EDRs. See "Configuring EDR Output Processing". When you configure the OUT_GenericStream module, you configure the EXT_OutFileManager module to specify file management options. See EXT_OutFileManager.
OUT_Realtime	The OUT_Realtime module converts data in the pipeline EDR output to flist format. See "Configuring a Real-Time Discounting Pipeline".

Table 80-1 (Cont.) Pipeline Manager Modules

Module	Description
OUT_Reject	Writes rejected EDRs to an output stream. The written record is exactly the same as the original input record. See "Configuring Output for Rejected or Duplicate EDRs". All registry entries and error messages are handled by the Output Collection module. See Output Collection. For more information, see the following documents: • About Standard Recycling • Handling Duplicate EDRs
Sequencer	Checks for duplicate CDR input files and adds tracking numbers to output streams.
Input Controller	Manages incoming input streams for its associated pipeline. See "Configuring EDR Input Processing".
Output Controller	Manages the output streams for its associated pipeline. See "Configuring EDR Output Processing".
Output Collection	Handles output streams. See "Configuring EDR Output Processing".
Pipeline Controller	Manages all processes in its associated pipeline. See Pipeline Controller.
Transaction Manager	Coordinates the state of all transactional modules and components in a pipeline.
Transaction ID Controller	Generates transaction IDs for all pipelines.
Transaction ID Database Generator	Stores transaction IDs in a Pipeline Manager database table.
Transaction ID File Generator	Stores transaction IDs in a file.

81
Pipeline Manager Function Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager function modules.

FCT_Account

The FCT_Account module adds customer data to an EDR.

It also does the following:

- Flags incoming CDRs to be suspended when the account is being rerated by **pin_rerate**.
- For exclusion rules for usage discounts, retrieves package list information from the DAT_AccountBatch module and adds this information to the PLAN_LIST block in the CustomerData block in the EDR. The package list includes all plans for the subscription service, including plans owned by any member services in the subscription group.
- For Balance Monitoring, retrieves the balance monitor information for an event owner from the DAT_AccountBatch module and fills the CustomerData block in the EDR with the monitor list.

Dependencies

Requires a connection to the DAT_AccountBatch module.

This module must run before the zoning and rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-1 lists the FCT_Account registry entries.

Table 81-1 FCT_Account Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_AccountBatch module.	Yes
DisableRatingProductCheck	Specifies whether the module rejects any CDRs with no rating charge offers. **True** = FCT_Account does not reject CDRs, if the configured event type for batch rating is not found in any of the charge offers owned by the service or account. **False** = FCT_Account rejects CDRs if the configured event type for batch rating is not found in any of the charge offers owned by the service or account.	No

Sample Registry

```
Account
{
  ModuleName = FCT_Account
  Module
  {
    Active         = True
    DataModule     = ifw.DataPool.CustomerData
    Offset         = 5
  }
}
```

Semaphore File Entries

Table 81-2 lists the FCT_Account Semaphore file entry.

Table 81-2 FCT_Account Semaphore File Entries

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CustomerSearch.Module.Active = False
```

EDR Container Fields

The FCT_Account module uses the EDR container fields listed in Table 81-3:

Table 81-3 FCT_Account EDR Container Fields

Alias field name Default field name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code that determines the EDR container fields that are used for the customer lookup. For example, a telephone service uses the A number to find the customer account.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the start timestamp of the event. The time zone information for this timestamp is stored in the field BDR_UTC_TIME_OFFSET.

Table 81-3 (Cont.) FCT_Account EDR Container Fields

Alias field name Default field name	Type	Access	Description
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC time zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/- *HHMM*.
DETAIL.CUST_A.ACTG_LAST_DATE	Date	Read	Contains the date that the current monthly cycle began.
DETAIL.CUST_A.ACTG_NEXT_DATE	Date	Read	Contains the date that the current monthly cycle ends.
DETAIL.CUST_A.ACTG_USED_DATE	Date	Write	Contains the date used for this EDR.
RESOURCE_LIST DETAIL.CUST_A.RESOURCE_LIST	String	Write	Contains a list of the balance elements included in the A-number customer's balance.
RESOURCE_LIST DETAIL.CUST_B.RESOURCE_LIST	String	Write	Contains a list of the balance elements included in the B-number customer's balance.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Create	Contains the charge offer rating indexes. This is a comma-separated list of all rating charge offers' indexes associated with the same service and event, and their priorities.
DETAIL.CUST_A.PRODUCT_PRIORITY	Integer	Read	Contains the priority for a charge offer.
DETAIL.CUST_A.PRODUCT.USAGE_START	Date	Read	Contains the start time for a charge offer.
DETAIL.CUST_A.PRODUCT.FIRST_USAGE_INDICATOR	Integer	Write	Specifies whether the account's charge offer is configured to start when first used and the state of the validity period.
DETAIL.CUST_A.DL.PD.FIRST_USAGE_INDICATOR	Integer	Write	Specifies whether the account's discount is configured to start when first used and the state of the validity period.
BALANCE_GROUP_ID DETAIL.CUST_A.ML.BALANCE_GROUP_ID	String	Create	Contains the balance monitor group ID.
MONITOR_OWNER_ACCT_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ACCT_ID	String	Create	Contains the monitor owner's account ID.
MONITOR_OWNER_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ID	String	Create	Contains the monitor owner ID.
MONITOR_OWNER_TYPE DETAIL.CUST_A.ML.MONITOR_OWNER_TYPE	String	Create	Contains the monitor owner type.

Table 81-3 (Cont.) FCT_Account EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.SHARED_PROFILE_LIST. ERA.LABEL	String	Write	Contains the label associated with a shared service profile. For example, MYFAMILY.
DETAIL.CUST_A.PRODUCT.ERA. LABEL	String	Write	Contains the label associated with an owned service profile. For example. MYFAMILY.
DETAIL.CUST_A. SHARED_PROFILE_LIST	Block	Write	Contains all the shared profiles, which the service shares as a member of one or more profile sharing groups.
DETAIL.CUST_A. SHARED_PROFILE_LIST.ERA	Block	Write	Contains shared ERA information.

Database Interface for the FCT_Account Module

The FCT_Account modules uses the following database tables:

- The FCT_Account and FCT_AccountRouter modules use the data in the IFW_ALIAS_MAP table to link an internal service code the EDR container field used for identifying an account.
- The FCT_Account and FCT_AccountRouter module use the data in the IFW_EDRC_DESC table to look up the alias mapping data. This table contains all valid EDR container fields for different format descriptions.
- The FCT_Account module uses the data in the IFW_EDRC_FIELD table to look up the alias mapping data. This table contains all valid EDR container fields for different format descriptions.
- The FCT_Account and FCT_AccountRouter use the data in the IFW_PIPELINE table to look up the alias mapping data. The IFW_PIPELINE database table defines the EDR formats that can be used for each pipeline.

FCT_AccountLPRouter

The FCT_AccountLPRouter module looks up the logical partition in which the account resides and writes it to the LOGICAL_PARTITION_ID EDR field.

Dependencies

Requires a connection to the DAT_AccountBatch module.

Registry Entries

Table 81-4 lists the FCT_AccountPRouter registry entries.

Table 81-4 FCT_AccountLPRouter Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_AccountBatch module.	Yes

Sample Registry

```
#------------------------------------------------
# AccountLPRouter
#------------------------------------------------
AccountLPRouterModule
{
  ModuleName = FCT_AccountLPRouter
  Module
  {
    Active       = True
    DataModule   = ifw.DataPool.CustomerData
  } # end Module
} # end LPRouter
#------------------------------------------------
```

Semaphore File Entries

Table 81-5 lists the FCT_AccountPRouter Semaphore file entry.

Table 81-5 FCT_AccountPRouter Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.AccountLPRouter.Module.Active = False
```

EDR Container Fields

The FCT_AccountLPRouter module uses the following EDR container fields:

Events

Table 81-6 lists the events for FCT_AccountPRouter.

Table 81-6 FCT_AccountPRouter Events

Alias Field Name Default Field Name	Type	Access	Description
LOGICAL_PARTITION_ID	String	Read	Contains the logical partition ID for the event.

FCT_AccountRouter

The FCT_AccountRouter module routes EDRs to the appropriate database schema in multischema systems. This module finds the customer's schema and routes the EDRs to the appropriate pipeline. See "Using Pipeline Manager with Multiple Database Schemas".

When used with standard recycling or Suspense Manager, this module routes call records from the pre-recycling pipeline to the appropriate rating pipeline.

> **Note:**
>
> FCT_AccountRouter runs in its own instance of Pipeline Manager and should be configured with its own registry file. Create a registry file that includes entries for the Input, FCT_AccountRouter, and Output modules.

Dependencies

Requires a connection to the DAT_AccountBatch module.

For general use, this module must run after the FCT_ServiceCodeMap module and before the rating modules.

For use with standard recycling or Suspense Manager using multiple database schemas, this module must be run before OUT_GenericStream in a pre-recycling pipeline.

This module sends output to a separate pipeline for each BRM database schema.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-7 lists the FCT_AccountRouter registry entries.

Table 81-7 FCT_AccountRouter Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_AccountBatch module.	Yes

Table 81-7 (Cont.) FCT_AccountRouter Registry Entries

Entry	Description	Mandatory
Mode	Specifies how this module routes EDRs to the appropriate pipeline. **ROUTER** = Routes EDRs based on the database schema ID. **RECYCLE** = Routes EDRs based on the pipeline name and schema ID. See "Configuring a Pre-Recycling Pipeline".	Yes
Streams	Lists the target output streams. The syntax for this section depends on whether the module is operating in **ROUTER** mode or **RECYCLE** mode. See "Configuring EDR Output Processing".	Yes

Sample Registries

Sample registry entries for the **ROUTER** mode:

```
AccountRouter
{
   ModuleName = FCT_AccountRouter
   Module
   {
      Active = True
      DataModule = ifw.DataPool.Account
      Mode=ROUTER
      Streams
      {
         1 = DB0001Stream
         2 = DB0002Stream
         3 = DB0003Stream
      }
   }
}
```

Sample registry entries for the **RECYCLE** mode:

```
AccountRouter
{
    ModuleName = FCT_AccountRouter
    Module
    {
        Active = True
        DataModule = ifw.DataPool.CustomerData
        Mode = RECYCLE

        Streams
        {
        # This section maps pipelines to their original stream
            1_Pipeline_A = StreamA_for_DB1
            1_Pipeline_B = StreamB_for_DB1
            2_Pipeline_C = StreamC_for_DB2
            2_Pipeline_D = StreamD_for_DB2

        # This section maps pipelines to their alternate stream
            1_Pipeline_C = StreamA_for_DB1
            1_Pipeline_D = StreamB_for_DB1
            2_Pipeline_A = StreamC_for_DB2
            2_Pipeline_B = StreamD_for_DB2
```

```
            # This section sends EDRs to the suspense stream
                0_* = SuspenseStream
            }
        }
}
```

Semaphore File Entries

Table 81-8 list the FCT_AccountRouter Semaphore file entry.

Table 81-8 FCT_AccountRouter Semaphore File Entry

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.Rating.Module.AccountRouter.Module.Active = False
```

EDR Container Fields

The FCT_AccountRouter module uses the EDR container fields listed in Table 81-9:

Table 81-9 FCT_AccountRouter EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code that determines the EDR container fields used for the customer lookup.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the start timestamp of the event. The time zone information for this timestamp is stored in the BDR_UTC_TIME_OFFSET field.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC time zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/- *HHMM*.

Database Tables

The FCT_Account module uses the following database tables:

- IFW_ALIAS_MAP. The FCT_Account and FCT_AccountRouter modules use the data in the IFW_ALIAS_MAP database table to link an internal service code the EDR container field used for identifying an account.
- IFW_EDRC_DESC. The FCT_Account and FCT_AccountRouter module use the data in the IFW_EDRC_DESC table to look up the alias mapping data. This table contains all valid

EDR container fields for different format descriptions. See "Using Pipeline Manager with Multiple Database Schemas".

- IFW_PIPELINE. The FCT_Account and FCT_AccountRouter use the data in this table to look up the alias mapping data. The IFW_PIPELINE database table defines the EDR formats that can be used for each pipeline. See "Using Pipeline Manager with Multiple Database Schemas".

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_AggreGate

The FCT_AggreGate module performs aggregation of data in EDR containers. See "Setting Up Pipeline Aggregation".

Dependencies

The FCT_AggreGate function module requires a connection to the DAT_ScenarioReader module.

This module runs after rating modules and can run in its own pipeline.

When you configure the FCT_CallAssembling function module to not drop EDRs from the pipeline, ensure that the FCT_AggreGate function module that counts them runs before the FCT_Reject function module. See "FCT_CallAssembling".

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-10 lists the FCT_AggreGate registry entries.

Table 81-10 FCT_AggreGate Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
BackOut	Specifies if the module is working in back-out mode, which is used for rerating. Possible values are **True** and **False**. Default = **False**	No
ControlFile	Specifies the definitions for the control file. The control files are used by the Database Loader utility to load the results into the database.	No

Table 81-10 (Cont.) FCT_AggreGate Registry Entries

Entry	Description	Mandatory
ControlFile.DataFilePath	Specifies whether the path to the data file is included in the control file. Possible values are **True** and **False**. Default = **True**	No
ControlFile.Suffix	Specifies the file name suffix for the control file. You can specify any suffix. Default = **.ctl**	No
IncludeCtlFile	Specifies whether to create control files. If **True**, control and data files are created. If **False**, only data files are created. Default = **True**	No
IncludeErrorEDRs	Specifies whether EDRs that include errors are included in the aggregation scenario. Possible values are **True** and **False**. Default = **False**	No
IncludeInvalidDetailEDRs	Specifies whether EDRs that are invalid are included in the aggregation scenario. Possible values are **True** and **False**. Default = **False**	No
ResultFile	Sub-entries define characteristics of the result file.	No
ResultFile.DoneSuffix	Specifies the file name suffix for processed files. You can specify any suffix. Default = **.dat**	No
ResultFile.TempSuffix	Specifies the file name suffix for temporary files created during processing. You can specify any suffix. Default = **.tmp**	No
ResultFile.WriteEmptyFile	Indicates whether to create an empty processed file if there are no processing results. Possible values are **True** and **False**. Default = **True**	No
ScenarioReaderDataModule	Specifies a connection to the DAT_ScenarioReader module.	Yes
Scenarios	Specifies the scenario that is processed and configured. See "Specifying Scenario Attributes". If nothing is entered, no processing occurs.	No
Scenarios	No value. Subentries define the scenarios to be processed and how they are configured. See "Specifying Scenario Attributes". If nothing is entered, no processing occurs.	No
Scenarios.*name*	Specifies the name of a scenario to be configured. The scenarios included with Revenue Assurance Manager have names such as **RA_01**, **RA_02** and so on.	Yes
Scenarios.*name***.ControlPointId**	Defines the Revenue Assurance control point that uses this scenario. Control point names must be unique system-wide. If a value is specified, the control point ID is included in the result file.	No
Scenarios.*name***.CtlDir**	Specifies the directory from which to read the control file. Use this entry when you want to override the default directory defined for the scenario.	No
Scenarios.*name***.DoneDir**	Specifies a path and name for processed files. Use this entry when you want to override the default directory.	No

Table 81-10 (Cont.) FCT_AggreGate Registry Entries

Entry	Description	Mandatory
Scenarios.*name*.FieldDelimiter	Specifies the delimiter of result fields. The default is a semicolon (**;**), which is the value defined in the IFW_SCENARIO table.	Yes
Scenarios.*name*.IncludeErrorEDRs	Speechifies whether EDRs that have errors are included in the aggregation processing. Possible values are **True** and **False**. Default value is **False**. **Note:** For Revenue Assurance, this parameter must be present and set to **True**.	No
Scenarios.*name*.IncludeInvalidDetailEDRs	Specifies whether invalid EDRs are included in aggregation processing. Possible values are **True** and **False**. Default value is **False**. **Note:** For Revenue Assurance, this parameter must be present and set to **True**.	No
Scenarios.*name*.IncludeProcessingTimestamps	Specifies whether transaction time range data is included in result files. Possible values are **True** and **False**. Default value is **False**.	No
Scenarios.*name*.TableName	Specifies the table used when DB Loader stores aggregation results in the database. The table name is also used for files created by this scenario. Use this entry when you want to override the default table name defined for the scenario.	No
Scenarios.*name*.TempDir	Specifies a path and name for a directory to use for temporary files created during processing. Use this entry when you want to override the default directory defined for the scenario.	No
Scenarios.*name*.Threshold	Specifies the maximum number of aggregations stored in memory before writing to disk. In most cases, there aren't enough aggregations to make the threshold meaningful. A typical value is a relatively large number, such as **99999**. Use this entry when you want to override the default directory defined for the scenario.	No

Sample Registry

```
Aggregate
{
  ModuleName = FCT_AggreGate
  Module
  {
    Active = TRUE

    ScenarioReaderDataModule = ifw.DataPool.ScenarioReader

    Scenarios
      {
      PURCHASE
      {
        TableName = PURCHASE_RESULT
        Threshold = 100000
        TempDir = result/temp
```

```
        DoneDir = result/done
        CtlDir = result/ctl
        FieldDelimiter = |
      }
      STAT
      {
        TempDir = result/temp
        DoneDir = result/done
        CtlDir = result/ctl
        }
        DAY
        {
      }
   }

    ResultFile
    {
      TempSuffix = .tmp
      DoneSuffix = .dat
      WriteEmptyFile = FALSE
    }

    ControlFile
      {
      IncludeCtlFile = FALSE
      Suffix = .ctl
      DataFilePath = TRUE
    }
  }
}
```

Semaphore File Entries

Table 81-11 lists the FCT_AggreGate Semaphore file entry.

Table 81-11 FCT_AggreGate Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Aggregate.Module.Active = False
```

EDR Container Fields

The module reads EDR container fields defined by the DAT_ScenarioReader module. See "DAT_ScenarioReader".

Events

Table 81-12 lists the FCT_AggreGate events.

Table 81-12 FCT_AggreGate Events

Event name	Trigger	Sender	Parameter
EVT_CTL_FILE_CREATED	Control file for the DB Loader utility was created.	DAT_ScenarioReader	File name

FCT_APN_Map

Before zoning: The FCT_APN_Map module maps the access point name (APN) to a physical PDP address.

After zoning: The FCT_APN_Map module enhances zone values to support enhanced zoning functionality.

Dependencies

Requires a connection to the Pipeline Manager database.

This module can be run before or after the zoning modules (FCT_Zone and FCT_PreRating).

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-13 lists the FCT_APN_Map registry entries.

Table 81-13 FCT_APN_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
APNGroup	Specifies the Access Point Name (APN) group value for the mapping. If you enter a group name, run the module before zoning. Otherwise, run it after zoning. You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes

Sample Registry

```
APN_Map
{
  ModuleName = FCT_APN_Map
  Module
  {
    Active = True
    APNGroup = apn_group
    DataConnection = integrate.DataPool.Login
  }
}
```

Semaphore File Entries

Table 81-14 lists the FCT_APN_Map Semaphore file entries.

Table 81-14 FCT_APN_Map Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
APNGroup	If you enter a group name, run the module before zoning. Otherwise, run it after zoning.
Reload	Reloads data from the database into memory.

Sample Semaphore File Entries

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.APNMap.Module.Active = True
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.APNMap.Module.APNGroup = apn_group
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.APNMap.Module.Reload{}
```

EDR Container Fields

The FCT_APN_Map module uses the EDR container fields listed in Table 81-15:

Table 81-15 FCT_APN_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
APN_ADDRESS DETAIL.ASS_GPRS_EXT.APN_ADDRESS	String	Read	Contains the access point name address.
ACTION_CODE DETAIL.ASS_GSMW_EXT.SS_PACKET.ACTION_CODE	String	Read	Contains the action code in supplementary service packet.
SS_EVENT DETAIL.ASS_GSMW_EXT.SS_PACKET.SS_EVENT	String	Read	Contains the supplementary service event.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Write	Contains the destination zone.
BDR_INTERN_APN_GROUP DETAIL.INTERN_APN_GROUP	String	Read	Contains the APN group.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read/Write	Contains the wholesale impact category.
RETAIL.IMPACT_CATEGORY DETAIL.RETAIL.IMPACT_CATEGORY	String	Read/Write	Contains the retail impact category.

Table 81-15 (Cont.) FCT_APN_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Write	Contains the impact category.
ASS_CBD_INTERN_APN_GROUP DETAIL.ASS_CBD.CP.INTERN_APN_GROUP	String	Read	Contains the APN group in the charge packet.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Read	Contains the wholesale or retail charge type. The default is retail.
ASS_ZBD_INTERN_APN_GROUP DETAIL.ASS_ZBD.ZP.INTERN_APN_GROUP	String	Read	Contains the APN group in zone breakdown records.
ASS_ZBD_ZONE_RESULT_VALUE_WS DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_WS	String	Write	Contains the wholesale zone result.
ASS_ZBD_ZONE_RESULT_VALUE_RT DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_RT	String	Write	Contains the retail zone result.

Database Tables

The FCT_APN_Map module uses the following database tables:

- IFW_APN_MAP. This table stores the APN mapping rules.
- IFW_APN_GROUP. The IFW_APN_GROUP table stores the APN groups used for APN mapping.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_ApplyBalance

The FCT_ApplyBalance module reads the discount packets added by DAT_Discount, adds the discounting sub-balance impact to the EDR, and updates the in-memory balance.

When the discount impacts a noncurrency balance element balance that starts on first usage, this module adds the validity period information to the EDR. See "About Setting the Validity of Balance Elements Impacted by Discounts ".

This module is mandatory when you configure batch discounting in Pipeline Manager. Add this module to the pipeline after the FCT_Rounding module.

Dependencies

Requires a connection to the DAT_BalanceBatch module.

This module should be in the same function pool as the FCT_Discount module for performance reasons and must run after that module.

This module also must run after FCT_Rounding.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-16 lists the FCT_ApplyBalance registry entries.

Table 81-16 FCT_ApplyBalance Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
BalanceDataModule	Specifies the connection to the DAT_BalanceBatch module.	Yes
DiscountDataModule	Specifies the connection to the DAT_Discount module.	Yes
FirstUsageCreateStream	Specifies the output stream for balance element balance impacts that whose validity periods start on first usage. See "About Setting the Validity of Balance Elements Impacted by Discounts ". Default = **FirstUsageResourceOutput**	No
IgnoreEDROnDeadlock	Specifies what the module should do when it encounters a deadlock. **True** = Ignore the EDRs and continue processing the EDR file. **False** = Roll back already processed EDRs and start reprocessing the same file.	No
NumberOfNotificationLimit	Specifies the maximum number of notification events that can be written into the output XML file. Once the maximum number of notification events is reached, the module creates another XML file.	No
NotificationOutputDirectory	Specifies the directory in which to write the output XML files.	No
NotificationOutputPrefix	Specifies the prefix of the output XML file.	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module.	Yes

Sample Registry

```
ApplyBalance
{
   ModuleName = FCT_ApplyBalance
   Module
   {
      Active                    = TRUE
      DiscountDataModule        = ifw.DataPool.DiscountModelDataModule
      BalanceDataModule         = ifw.DataPool.BalanceDataModule
      PortalConfigDataModule    = ifw.DataPool.PortalConfigDataModule
      IgnoreEDROnDeadlock       = False

     NumberOfNotificationLimit = ifw.DataPool.PortalConfigDataModule
     NotificationOutputDirectory           = ./data/out/notifybalancebreach
     NotificationOutputPrefix              = balancebreach_
   }
}
```

Semaphore File Entries

Table 81-17 lists the FCT_ApplyBalance Semaphore file entries.

Table 81-17 FCT_ApplyBalance Semaphore File Entries

Entry	Description
ReloadCreditThresholdParam	Reloads the value from the **CreditThresholdChecking** business parameter.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApplyBalanceModule.Module.Reload
CreditThresholdParam{}
```

EDR Container Fields

Table 81-18 lists the FCT_ApplyBalance EDR container fields.

Table 81-18 FCT_ApplyBalance EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.EVENT_TYPE	String	Read	Specifies the BRM event type.
INTERNAL.TRANSACTION_ID	String	Read	Specifies the transaction ID. This is used for queuing.
DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Specifies the process status. If set to 2, a recycle test is in progress, and this container is skipped.
DETAIL.ASS_CBD.DP.OBJECT_ACCOUNT	Integer	Read	Specifies the POID of the discount offer owner.
DETAIL.ASS_CBD.DP.OFFERING_POID	String	Read	Specifies the POID of the account's purchased discount offer.
DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Read	Specifies the ID of the balance element impacted by the discount.
DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read	Specifies the discount grant amount. This can be currency or noncurrency.
DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	String	Read	Specifies the POID of the account's balance group that is impacted by the discount.
DETAIL.ASS_CBD.DP.VALID_FROM	Date	Read	Specifies the date when the discount becomes valid.
DETAIL.ASS_CBD.DP.VALID_FROM_DETAIL	Integer	Read	Specifies the mode of the discounts' start time (such as first-usage or relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.
DETAIL.ASS_CBD.DP.VALID_TO	Date	Read	Specifies the date when the discount is no longer valid.

Table 81-18 (Cont.) FCT_ApplyBalance EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_CBD.DP.VALID_TO_DETAIL	Integer	Read	Specifies the mode of the discounts' end time (such as relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.
DETAIL.ASS_CBD.DP.SUB_BALANCE	Packet	Write	Specifies the sub-balance that is impacted by the discount.
DETAIL.ASS_CBD.DP.SUB_BALANCE.REC_ID	Integer	Write	Specifies the ID of the sub-balance impacted by this discount.
DETAIL.ASS_CBD.DP.SUB_BALANCE.AMOUNT	Decimal	Write	Specifies the amount of the sub-balance impacted by this discount packet.
DETAIL.ASS_CBD.DP.SUB_BALANCE.GRANTOR	String	Write	Specifies the charge offer or discount that granted this balance element.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM	Date	Write	Specifies the date when this sub-balance becomes valid.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO	Date	Write	Specifies the date when this sub-balance is no longer valid.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM_DETAILS	Integer	Write	Specifies the mode of the sub-balance validity period and the relative offset start time details.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO_DETAILS	Integer	Write	Specifies the mode of the sub-balance validity period and the relative offset end time details.
DETAIL.ASS_CBD.DP.SUB_BALANCE.CONTRIBUTOR	String	Write	Specifies the service or account that contributes to the sub-balance amount.
DETAIL.ASS_CBD.UBP	Packet	Write	The update balance packet. This packet contains validity period information for all the account's sub-balances that start on first usage. This packet is added when a sub-balance with a first-usage start time is consumed for the first time.
DETAIL.ASS_CBD.UBP.BALANCE_GROUP_ID	Integer	Write	Specifies the POID of the account's balance group associated with a balance element balance that starts on first usage.
DETAIL.ASS_CBD.UBP.RESOURCE_ID	Integer	Write	Specifies the ID of the associated balance element.
DETAIL.ASS_CBD.UBP.RECORD_NUMBER	Integer	Write	Specifies the sequence number of the record in the file.
DETAIL.ASS_CBD.UBP.VALID_FROM	Date	Write	Specifies the start time of the balance element balance.
DETAIL.ASS_CBD.UBP.VALID_TO	Date	Write	Specifies the end time of the balance element balance.

Table 81-18 (Cont.) FCT_ApplyBalance EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_CBD.UBP.VALID_FROM_DETAIL	Integer	Write	Specifies the balance element balance start time details: the mode of the validity period (such as first-usage or relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.
DETAIL.ASS_CBD.UBP.VALID_TO_DETAIL	Integer	Write	Specifies the balance element balance end time details: the mode of the validity period (such as relative), the relative offset unit (such as minutes, months, or cycles) and the number of offset units.

FCT_BatchSuspense

This module is used by the Suspense Manager service to handle file-level suspense operations. It generates the suspended batch create and update streams to be loaded into the BRM database for suspense batch files.

> **Note:**
>
> This module must be placed before all the validation modules/iScripts in a pipeline.

This module also adds suspense reason and suspense subreason codes to batches. See:

- Setting Up Suspended Batch (SB) Loader for Suspense Manager
- About the FCT_BatchSuspense Module
- About Suspense Manager

Dependencies

Requires a connection to the BRM database.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-19 lists the FCT_BatchSuspense registry entries.

Table 81-19 FCT_BatchSuspense Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
ResubmitDataModule	Specifies a connection to the DAT_Resubmit module.	Yes
DataConnection	Specifies the connection to the BRM database.	Yes
PipelineCategory	Specifies the pipeline category. Default = **CDRPipeline**	Yes
StorableClass	Specifies the storable class used to store suspended batch records. Default =/suspended_batch/cdr See "Suspending CDR Files".	Yes (for Batch file processing)
SuspenseFile	Specifies the batch suspense file this module generates: • **Path** Specifies the path where the data file will be written to. • **Prefix** Specifies the prefix for the resulting filename. • **Suffix** Specifies the suffix for the resulting filename. • **TempDataPrefix** Specifies the prefix for the temporary filename that is used while the file is being built.	Yes

Sample Registry

```
#---------------------------
# Batch Suspense FCT
#---------------------------
BatchSuspense
{
  ModuleName                = FCT_BatchSuspense
  Module
  {
    Active                  = TRUE
    ResubmitDataModule      = ifw.DataPool.ResubmitBatch
    DataConnection          = ifw.DataPool.LoginInfranet
    PipelineCategory        = CDRPipeline
    StorableClass           = /suspended_batch/cdr
    SuspenseFile
    {
       Path = ..
       Prefix = Framework
       Suffix = msg
       TempDataPrefix = rej_
          }
  }
}
```

EDR Container Fields

Table 81-20 lists the FCT_BatchSuspense EDR container fields.

Table 81-20 FCT_BatchSuspense EDR Container Fields

Alias Field Name Default Field Name	Type	Description
BATCH_ID HEADER.BATCH_ID	String	The unique identifier for the batch file. The batch file's BATCH_ID field is set in the HEADER block when it is received by the pipeline and this field receives its value from that field.
SEQUENCE_NUMBER HEADER.SEQUENCE_NUMBER	String	The suspense sequence number.
SENDER HEADER.SENDER	String	The sender.
TAP_PROCESSING_INFO HEADER.TAP_PROCESSING_INFO	String	Tap specific information (e.g TAP file name for a specific RAP etc). The format of this field is specific to TAP/RAP processing modules and this information created and interpreted by these modules
OVERRIDE_REASONS HEADER.OVERRIDE_REASONS	String	Contains the batch suspense reasons that are overridden from SMC while resubmitting the batch. Mapped from the error code. Used by Suspense Manager.

FCT_BillingRecord

The FCT_BillingRecord module consolidates balance impact data into an associated BRM billing record and one or more balance impact packets. See "About Consolidation for BRM Billing".

> **Note:**
>
> The FCT_ItemAssign module handles items assigned for usage events.

> **Note:**
>
> Don't use the FCT_BillingRecord module in a CIBER roaming revenue assurance environments. For more information, see "Billing Consolidation with CIBER Roaming and Revenue Assurance".

Dependencies

Requires a connection to the following modules:

- DAT_AccountBatch
- DAT_BalanceBatch

- DAT_Currency
- DAT_ItemAssign

This module must run after the FCT_MainRating and FCT_Discount modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-21 lists the FCT_BillingRecord registry entries.

Table 81-21 FCT_BillingRecord Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **TRUE** = Active **FALSE** = Inactive You can use this entry in a semaphore file.	Yes
AccountDataModule	Specifies a connection to the DAT_AccountBatch module.	Yes
BalanceDataModule	Specifies a connection to the DAT_BalanceBatch module.	Yes
ChargeBreakDownRecordType	By default, FCT_BillingRecord module processes Charge Breakdown records whose record type is 981. Use this entry to specify additional Charge Breakdown records. **Note:** You can specify multiple Charge Breakdown record types. Separate each record type by using a comma.	No
CurrencyDataModule	Specifies a connection to the DAT_Currency module.	Yes
CurrencyType	Specifies the CHARGED_CURRENCY_TYPE value that the charge packets should contain. The possible values are: • **H** = Home currency • **B** = Billing currency • **R** = Rating currency Default = **H**.	No
ItemAssignDataModule	Specifies a connection to the DAT_ItemAssign module.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module.	Yes
RatingPipeline	Specifies whether the module is running in the rating or rerating pipeline: **FALSE** = Rerating **TRUE** = Rating	No
RatePlanType	By default, FCT_BillingRecord processes charge packets whose rateplan type is R. Use this entry to specify additional rateplan types.	No
SetTaxLocales	Specifies whether to view the resulting tax locales in the **PIN_TAX_LOCALES** field: **FALSE** = Viewing the resulting tax locales in the **PIN_TAX_LOCALES** field is disabled. **TRUE** = View the resulting tax locales in the **PIN_TAX_LOCALES** field; for example, phone numbers. Default = **FALSE**.	No

Sample Registry Entry

```
AddInfranetBillingRecord
{
  ModuleName = FCT_BillingRecord
  Module
  {
    Active = TRUE
    AccountDataModule  = ifw.DataPool.CustomerData
    PortalConfigDataModule   = ifw.DataPool.PortalConfigDataModule
    BalanceDataModule = ifw.DataPool.BalanceDataModule
    ChargeBreakDownRecordType = 981
    CurrencyType = R
    CurrencyDataModule = ifw.DataPool.CurrencyDataModule
    ItemAssignDataModule = ifw.DataPool.ItemAssignDataModule
    RatingPipeline = TRUE
    RatePlanType = W
    SetTaxLocales = FALSE
  }
}
```

Semaphore File Entries

Table 81-22 lists the FCT_BillingRecord Sempahore file entries.

Table 81-22 FCT_BillingRecord Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.AddInfranetBillingRecord.
Module.Active = false
```

EDR Container Fields

Table 81-23 lists the FCT_BillingRecord EDR container fields.

Table 81-23 FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the normalized A number.
ACCOUNT_POID DETAIL.ASS_PIN.ACCOUNT_POID	String	Write	Contains the resulting account POID.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ACCOUNT_POID DETAIL.ASS_PIN.MP.ACCOUNT_POID	String	Create	POID of the account that the balance impact applies to. **Derivation:** Mandatory. Calculated.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Read	Contains the record type of the associated charge record.
ASS_CBD_RUM_NAME DETAIL.ASS_CBD.RUM_NAME	String	Read	Contains the RUM name.
ASS_PIN_BALANCE_PACKET DETAIL.ASS_PIN.NUMBER_OF_BALANCE_PACKETS	Integer	Write	Contains the resulting number of balance packets.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the normalized B number.
BAL_GRP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	Read	Contains the balance group POID of the service.
BAL_GRP_POID DETAIL.ASS_PIN.MP.BAL_GRP_POID	String	Create	Balance monitor group that the balance impact applies to. **Derivation:** Mandatory. Calculated.
BALANCE_GROUP_ID DETAIL.CUST_A.ML.BALANCE_GROUP_ID	String	Read	Balance monitor group ID.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC offset.
BP_ACCOUNT_POID DETAIL.ASS_PIN.BP.ACCOUNT_POID	String	Write	Contains account POID.
BP_RUM_ID DETAIL.ASS_PIN.BP.RUM_ID	Long	Read/Write	Contains RUM id of the balance packet.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read	Contains the currency.
CHARGED_AMOUNT_VALUE_ORIG DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE_ORIG	Decimal	Read	Contains the charged amount value.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read	Contains the charged amount value.
CHARGED_CURRENCY_TYPE DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	Read	Contains the currency type: • **H** = Home • **R** = Rating • **B** = Billing
CHARGED_TAX_CODE DETAIL.ASS_CBD.CP.CHARGED_TAX_CODE	String	Read	Contains the charged tax code.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	Read	Contains charging end time.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains charging start time.
CONNECT_SUBTYPE DETAIL.CONNECT_SUBTYPE	String	Read	Contains the connect subtype.
DETAIL.ASS_CBD.CP.ITEM_TAG	String	Read	Contains the name of the charge item for balance impacts.
DETAIL.ASS_CBD.CP.ITEM_POID	String	Read	Contains the POID of the associated charge item for balance impacts.
DETAIL.ASS_CBD.DP.ITEM_TAG	String	Read	Contains the name of the discount item for balance impacts.
DETAIL.ASS_CBD.DP.ITEM_POID	String	Read	Contains the POID of the associated discount item for balance impacts.
DETAIL.ASS_CBD.TP.ITEM_TAG	String	Read	Contains the name of the tax item for balance impacts.
DETAIL.ASS_CBD.TP.ITEM_POID	String	Read	Contains the POID of the associated tax item for balance impacts.
CP_RUM_ID DETAIL.ASS_CBD.CP.RUM_ID	Long	Read	Contains RUM id.
DETAIL.ASS_CBD.CP.RECORD_TYPE	String	Read	Contains charge packet record type.
DETAIL.ASS_CBD.DP	Record	Read	The discount packet record.
DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	Integer	Read	Contains the ID of the balance group that the discount applies to.
DETAIL.ASS_CBD.DP.GLID	String	Read	Contains the G/L ID.
DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read	Contains the discount amount granted. This can be currency or noncurrency.
DETAIL.ASS_CBD.DP.GRANTED_QUANTITY	Decimal	Read	Contains the discount base values used to calculate the amount granted.
DETAIL.ASS_CBD.DP.IMPACT_CATEGORY	String	Read	Contains the discount impact category.
DETAIL.ASS_CBD.DP.NODE_LOCATION	String	Read	Contains the node location of the discount.
DETAIL.ASS_CBD.DP.OBJECT_ACCOUNT	String	Read	Contains the POID of the discount owner.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_CBD.DP.OBJECT_ID	String	Read	Contains the ID of the discount or sponsor object.
DETAIL.ASS_CBD.DP.OBJECT_OWNER_ID	String	Read	Contains the POID type of the discount owner.
DETAIL.ASS_CBD.DP.OBJECT_OWNER_TYPE	String	Read	Contains the POID type of the discount owner.
DETAIL.ASS_CBD.DP.OBJECT_TYPE	String	Read	Contains the discount or sponsor object that generated the discount.
DETAIL.ASS_CBD.DP.RATETAG	String	Read	Contains the rate tag for the discount.
DETAIL.ASS_CBD.DP.RESOURCE_ID	String	Read	Contains the ID of the balance element impacted.
DETAIL.ASS_CBD.DP.SUB_BALANCE.AMOUNT	Decimal	Read	Contains the amount applied to this sub-balance.
DETAIL.ASS_CBD.DP.SUB_BALANCE.CONTRIBUTOR	String	Read	Contains the sub-balance contributor field.
DETAIL.ASS_CBD.DP.SUB_BALANCE.REC_ID	Integer	Read	Contains the record ID of the sub-balance record.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM	Date	Read	Contains the date the balance element is valid from.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO	Date	Read	Contains the date the balance element is valid to.
DETAIL.ASS_CBD.DP.TAX_CODE	String	Read	Contains the tax code for the discount.
DETAIL.ASS_PIN.BI.PIN_RATE_TAG	String	Write	Contains the name of the charge that provided a promotional savings.
DETAIL.ASS_PIN.BP.BAL_GRP_POID	String	Write	Contains the POID of the balance group that is impacted.
DETAIL.ASS_PIN.BP.ITEM_POID	String	Write	Contains the POID of the associated item.
DETAIL.ASS_PIN.SBI	Record	Write	The sub-balance impact record.
DETAIL.ASS_PIN.SBI.BAL_GRP_POID	String	Write	Contains the POID of the balance group that is impacted.
DETAIL.ASS_PIN.SBI.PIN_RESOURCE_ID	Integer	Write	Contains the balance element ID.
DETAIL.ASS_PIN.SBI.RECORD_NUMBER	Integer	Write	Contains the record number of the sub-balance impact record.
DETAIL.ASS_PIN.SBI.RECORD_TYPE	String	Write	Contains the record type of the sub-balance impact record.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ASS_PIN.SBI.SB	Record	Write	The sub-balance record.
DETAIL.ASS_PIN.SBI.SB.CONTRIBUTOR	String	Write	Contains the sub-balance contributor field.
DETAIL.ASS_PIN.SBI.SB.PIN_AMOUNT	Decimal	Write	Contains the amount of the sub-balance impact.
DETAIL.ASS_PIN.SBI.SB.RECORD_NUMBER	Integer	Write	Contains the record number of the sub-balance record.
DETAIL.ASS_PIN.SBI.SB.RECORD_TYPE	String	Write	Contains the record type of the sub-balance record.
DETAIL.ASS_PIN.SBI.SB.VALID_FROM	Date	Write	Contains the date the sub-balance balance element is valid.
DETAIL.ASS_PIN.SBI.SB.VALID_TO	Date	Write	Contains the date the sub-balance balance element is no longer valid.
DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Contains the BRM account ID.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Long	Read	Contains the index of the purchased charge offer.
DETAIL.CUST_A.PRODUCT.OFFERING_POID	String	Read	Contains the charge offer's node location.
DETAIL.CUST_A.PRODUCT.PRODUCT_ID	String	Read	Contains the BRM charge offer ID.
DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Contains the ID of the charge offer.
DETAIL.CUST_A.PRODUCT.SERVICE_TYPE	String	Read	Contains the service type of the charge offer.
DETAIL.CUST_A.PRODUCT.SERVICE_USED_ITEM_POID	String	Read	Contains the item POID.
DETAIL_OBJECT_CACHE_TYPE DETAIL.OBJECT_CACHE_TYPE	Long	Write	Contains the cache type of the associated object.
DP_AMOUNT_ORIG DETAIL.ASS_CBD.DP.GRANTED_AMOUNT_ORIG	Decimal	Read	Contains original discount amount
DP_INTERN_RUM_ID DETAIL.ASS_CBD.DP.INTERN_RUM_ID	Long	Read	Contains discount RUM id.
DP_OFFERING_POID DETAIL.ASS_CBD.DP.OFFERING_POID	String	Read	Contains discount offering POID.
DP_RESOURCE_ID_ORIG DETAIL.ASS_CBD.DP.RESOURCE_ID_ORIG	Long	Read	Contains original balance element ID.
DP_SB_GRANTOR DETAIL.ASS_CBD.DP.SUB_BALANCE.GRANTOR	String	Read	Contains discount sub-balance grantor.
DP_SB_VALID_FROM_DETAILS DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM_DETAILS	Long	Read	Contains discount sub-balance valid from details.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DP_SB_VALID_TO_DETAILS DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO_DETAILS	Long	Read	Contains discount sub-balance valid to details.
EXCHANGE_CURRENCY DETAIL.ASS_CBD.CP.EXCHANGE_CURRENCY	String	Read	Contains currency for exchange.
EXPIRED_UNITS DETAIL.ASS_CBD.CP.DP. EXPIRED_UNITS	String	Write	Contains the number of units that exceed the maximum.
GRANTED_DISCOUNT_AMOUNT_VALUE DETAIL.ASS_CBD.CP.GRANTED_DISCOUNT_AMOUNT_VALUE	Decimal	Read	Contains the granted discount amount value.
GRANTOR DETAIL.ASS_PIN.SBI.SB.GRANTOR	String	Write	Contains grantor of sub-balance impact.
ITEM_POID DETAIL.ASS_PIN.ITEM_POID	String	Write	Contains the resulting item POID.
ITEM_TAG DETAIL.ITEM_TAG	String	Read	Contains the item tag.
MONITOR_OWNER_ACCT_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ACCT_ID	String	Read	Monitor owner's account ID.
MONITOR_OWNER_ID DETAIL.CUST_A.ML.MONITOR_OWNER_ID	String	Read	Monitor owner ID.
MONITOR_OWNER_TYPE DETAIL.CUST_A.ML.MONITOR_OWNER_TYPE	String	Read	Monitor owner type.
MONITOR_SUB_BAL DETAIL.ASS_PIN.MSBI.MONITOR_SUB_BAL	SB	Create	Sub-balance monitored.
MSBI_BAL_GRP_POID DETAIL.ASS_PIN.MSBI.BAL_GRP_POID	String	create	Contains balance group POID of monitor sub-balance impact.
MSBI_MSB_PIN_AMOUNT DETAIL.ASS_PIN.MSBI.MSB.PIN_AMOUNT	Decimal	Write	Contains amount of monitor sub-balance.
MSBI_MSB_RECORD_NUMBER DETAIL.ASS_PIN.MSBI.MSB.RECORD_NUMBER	Integer	Write	Contains record number of monitor sub-balance.
MSBI_MSB_RECORD_TYPE DETAIL.ASS_PIN.MSBI.MSB.RECORD_TYPE	String	Write	Contains record type of monitor sub-balance.
MSBI_MSB_VALID_FROM DETAIL.ASS_PIN.MSBI.MSB.VALID_FROM	Date	Write	Contains valid from of monitor sub-balance.
MSBI_MSB_VALID_TO DETAIL.ASS_PIN.MSBI.MSB.VALID_TO	Date	Write	Contains valid to of monitor sub-balance.
MSBI_PIN_RESOURCE_ID DETAIL.ASS_PIN.MSBI.PIN_RESOURCE_ID	Long	Write	Contains balance element id of monitor sub-balance impact.
MSBI_RECORD_TYPE DETAIL.ASS_PIN.MSBI.RECORD_TYPE	String	Write	Contains record type of monitor sub-balance impact.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
NET_QUANTITY DETAIL.NET_QUANTITY	Decimal	Write	Contains net quantity.
OBJECT_CACHE_TYPE DETAIL.ASS_PIN.BP.OBJECT_CACHE_TYPE"	Integer	Write	Contains the cache type of the associated object. **Possible values:** • **2** (POSTPAID) • **0** (CONVERGENT).
ORIGINAL_EVENT_POID DETAIL.ASS_PIN.ORIGINAL_EVENT_POID	String	Read	Contains original event POID.
PIN_AMOUNT_DEFERRED DETAIL.ASS_PIN.BP.PIN_AMOUNT_DEFERRED	Integer	Write	Specifies whether an EDR contains tax data. This field is set to **0** when an EDR contains a tax packet and a PIN_AMOUNT when an EDR doesn't contain a tax packet.
PIN_AMOUNT_ORIG DETAIL.ASS_PIN.BP.PIN_AMOUNT_ORIG	Decimal	Read/Write	Contains original amount.
PIN_AMOUNT DETAIL.ASS_PIN.BP.PIN_AMOUNT	Decimal	Write	Contains the resulting amount used to update BRM balances.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
PIN_AMOUNT DETAIL.ASS_PIN.MP.PIN_AMOUNT	9(11)	Create	Amount of one balance element impact for the account balance. The value can be either positive or negative. The value is added to the PIN_FLD_CURRENT_BAL field of the PIN_FLD_BALANCES array in the account object specified by PIN_FLD_ACCOUNT_OBJ. **Note:** In case of multiple-RUM rating, this value might be a total value. Possible values: Price. See below for minimum and maximum values.If no price is given, this is a space, for example, NULL in a database. The format is variable floating point. The floating decimal point must be set if the given value is not in the required format. Maximum value: 99999999999 Minimum value: -9999999999 Examples: • '00000000125' for 125,00 • '00000012.50' for 12,50 • '-0012.56780' for -12,5678 Derivation: Mandatory. Derived from the object, **/event/PIN_FLD_BAL_IMPACTS.PIN_FLD_AMOUNT**. The post-mapping processor decides what mapping rule applies to this attribute; for example, add multiple charge packet values. **Note:** This value does not include any granted discounts.
PIN_DISCOUNT DETAIL.ASS_PIN.BP.PIN_DISCOUNT	Decimal	Write	Contains the resulting discount values.
PIN_GL_ID DETAIL.ASS_PIN.BP.PIN_GL_ID	String	Write	Contains the resulting G/L ID.
PIN_IMPACT_CATEGORY DETAIL.ASS_PIN.BP.PIN_IMPACT_CATEGORY	String	Write	Contains the impact category from the charge packet or the discount packet, depending on which was passed.
PIN_IMPACT_TYP DETAIL.ASS_PIN.BP.PIN_IMPACT_TYPE	Long	Write	Contains impact type.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
PIN_INFO_STRING DETAIL.ASS_PIN.BP.PIN_INFO_STRING	String	Read/Write	Contains information of balance packet.
PIN_INVOICE_DATA DETAIL.ASS_PIN.PIN_INVOICE_DATA	String	Write	Contains the resulting BRM invoice data.
PIN_OFFERING_POID DETAIL.ASS_PIN.BP.PIN_OFFERING_POID	String	Write	Uniquely identifies an account charge offer, discount, or sponsor.
PIN_PRODUCT_POID DETAIL.ASS_PIN.BP.PIN_PRODUCT_POID	String	Write	Contains the resulting charge offer object.
PIN_QUANTITY DETAIL.ASS_PIN.BP.PIN_QUANTITY	Decimal	Write	Contains the resulting quantity used to update BRM accounts.
PIN_RATE_TAG DETAIL.ASS_PIN.BP.PIN_RATE_TAG	String	Write	Contains the rate tag from the discount packet.
PIN_RESOURCE_ID_ORIG DETAIL.ASS_PIN.BP.PIN_RESOURCE_ID_ORIG	Long	Write	Contains original balance element id.
PIN_RESOURCE_ID DETAIL.ASS_PIN.BP.PIN_RESOURCE_ID	String	Write	Contains the resulting ID of the BRM balance element.
PIN_RESOURCE_ID DETAIL.ASS_PIN.MP.PIN_RESOURCE_ID	9(9)	Create	Numeric value of the balance element that is impacted; for example, 840 for US dollars. Possible value: Any configured BRM balance element ID. **Derivation:** Mandatory.
PIN_TAX_CODE DETAIL.ASS_PIN.BP.PIN_TAX_CODE	String	Write	Contains the resulting BRM tax code.
PRICEMODEL_TYPE DETAIL.ASS_CBD.CP.PRICEMODEL_TYPE	String	Read	Contains the pricing type.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read	Contains rateplan code.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Read	Contains the charge type.
RECORD_TYPE DETAIL.ASS_PIN.MP.RECORD_TYPE	String	Create	Extended to be 3 bytes long. Possible values: • 800: monitor packet • 805: monitor sub-balance impact • 807: monitor sub-balance
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Long	Read	Contains charge packet balance element ID.
RESOURCE_ORIG DETAIL.ASS_CBD.CP.RESOURCE_ID_ORIG	Long	Read	Contains charge packet original balance element ID.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RM_NET_QUANTITY DETAIL.ASS_CBD.RM.NET_QUANTITY	Decimal	Read/Write	Contains net quantity of RUM.
ROUNDED_QUANTITY_VALUE DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_VALUE	Decimal	Read	Contains the rounded quantity value that was used for the price calculations.
RUM_NAME DETAIL.ASS_PIN.RUM_NAME	String	Write	Contains RUM name.
SERVICE_POID DETAIL.ASS_PIN.SERVICE_POID	String	Write	Contains the resulting service POID.
TAX_LOCALES DETAIL.ASS_PIN.PIN_TAX_LOCALES	String	Write	Contains the resulting tax locales string.
TJ_PIN_AMOUNT DETAIL.ASS_PIN.BP.TJ.PIN_AMOUNT	Decimal	create	Contains amount of tax jurisdiction of balance packet.
TJ_PIN_TAX_RATE DETAIL.ASS_PIN.BP.TJ.PIN_TAX_RATE	String	Create	Contains tax rate of tax jurisdiction of balance packet.
TJ_PIN_TAX_TYPE DETAIL.ASS_PIN.BP.TJ.PIN_TAX_TYPE	String	Read/create	Contains tax type of tax jurisdiction of balance packet.
TJ_PIN_TAX_VALUE DETAIL.ASS_PIN.BP.TJ.PIN_TAX_VALUE	Decimal	Create	Contains tax value of tax jurisdiction of balance packet.
TJ_RECORD_TYPE DETAIL.ASS_PIN.BP.TJ.RECORD_TYPE	String	Create	Contains record type of tax jurisdiction of balance packet.
TP_RELATED_CHARGE_INFO_ID DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Long	Read	Contains related charge info id.
TP_TAX_CODE DETAIL.ASS_CBD.TP.TAX_CODE	String	Read	Contains tax code.
TP_TAX_RATE DETAIL.ASS_CBD.TP.TAX_RAT	String	Read	Contains tax rate.
TP_TAX_TYPE DETAIL.ASS_CBD.TP.TAX_TYPE	String	Read	Contains tax type.
TP_TAX_VALUE_ORIG DETAIL.ASS_CBD.TP.TAX_VALUE_ORIG	Decimal	Read	Contains original tax value.
TP_TAX_VALUE DETAIL.ASS_CBD.TP.TAX_VALUE	Decimal	Read	Contains tax value.
TP_TAXABLE_AMOUNT DETAIL.ASS_CBD.TP.TAXABLE_AMOUNT	Decimal	Read	Contains taxable amount.
USAGE_GL_ACCOUNT_CODE DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Read	Contains G/L account code.
VALID_FROM_DETAILS DETAIL.ASS_PIN.SBI.SB.VALID_FROM_DETAILS	Long	Write	Contains valid from details of sub-balance impact.

Table 81-23 (Cont.) FCT_BillingRecord EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
VALID_FROM DETAIL.ASS_PIN.MSB.VALID_FROM	Date	Create	Valid from date for this sub-balance.
VALID_TO_DETAILS DETAIL.ASS_PIN.SBI.SB.VALID_TO_DETAILS	Long	Write	Contains valid to details of sub-balance impact.
VALID_TO DETAIL.ASS_PIN.MSB.VALID_TO	Date	Create	Valid-to date for this sub-balance.

FCT_CallAssembling

The FCT_CallAssembling module assembles the multiple CDRs that comprise a single wireless call into a single EDR that Pipeline Manager can process. See "Assembling EDRs".

Dependencies

You must run this module early in a pipeline to assemble EDRs. You must run it before FCT_Discard.

When you configure the FCT_CallAssembling function module to not drop EDRs from the pipeline, ensure that the FCT_AggreGate function module that counts them runs before the FCT_Reject function module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-24 lists the FCT_CallAssembling registry entries.

Table 81-24 FCT_CallAssembling Registry Entries

Entry	Description	Mandatory
Active	Turns FCT_CallAssembling module processing on and off. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
AssembledEDR	Specifies a list of fields that the EDR takes from the **L** call segment and appends it to the last EDR (if the two are different). No default entry. See "Capturing Fields From the Last Call Record ".	No
AssembleSGSN	Turns SGSN data capture on and off. If **True**, this entry waits for all CDRs to arrive before rating a call. If your system does not process TAP records, leave this set to **False** to save system resources. **True** = SGSN data recorded **False** = SGSN data not recorded Default = **False** See "Rating Calls by Volume of Data Sent ".	No

Table 81-24 (Cont.) FCT_CallAssembling Registry Entries

Entry	Description	Mandatory
AssembleVolume	Turns volume rating on and off. If **True**, this entry waits for all CDRs to arrive before rating a call. If your system does not require volume rating, leave this set to **False** to save system resources. **True** = volume rating on **False** = volume rating off Default = **False** See "Rating Calls by Volume of Data Sent ".	No
CallDurationTolerance	Specifies an allowable cumulative time error for a single call (in seconds). Used with **SplitAtGaps = True**. Default = **60** See "Specifying a Time Error".	No
DropLateCDRs	Specifies how to handle the output of late EDRs: • **True** = Drop late EDRs from the pipeline. • **False** = Send late EDRs through the pipeline as non-valid. Default = **True** See "Dropping Late Calls".	No
EmitPartialEDROnUpgrade	Specifies the results of the **UpgradeFlushLimit** semaphore. Results are one of the following: • Silently drops EDRs from the in-memory **.dat** file. • Emit partial EDRs for revenue assurance tracking. (Partial EDRs should be sent to the discard stream.) Default = **False** (disabled)	No
FileName	Specifies the base file name for the data files. The transaction ID and the suffix are appended. See "Managing the Call Assembling Data Files".	Yes
MaxDuration	Directs FCT_CallAssembling to rate segments of a wireless call periodically. This entry specifies the maximum amount of time (in seconds) that a call can remain open before FCT_CallAssembling rates the segments that have arrived. This module recalculates the call duration for every call each time a new call segment arrives and compares it to the **MaxDuration** setting. If the new time duration equals or exceeds the setting for **MaxDuration,** FCT_CallAssembling emits an EDR to rate the existing portion of the call. For details and a comparison to **FlushLimit**, see "Rating Calls by Time Duration". No default entry.	No
Mode	Change this entry *only* if you are creating data upgrade pipelines that are used when changing an EDR container description. The possible values are: **Normal**. The default mode, and the most common use of this module. Directs FCT_CallAssembling to assemble CDRs into EDRs so Pipeline Manager can process them. **RestoreEDRs**. Directs FCT_CallAssembling to read serialized EDRs in sequence from data files and inserts them into the pipeline. **UpGradeData**. Directs FCT_CallAssembling to update data files based on the EDRs it receives. Default = **Normal**	Yes

Table 81-24 (Cont.) FCT_CallAssembling Registry Entries

Entry	Description	Mandatory
Path	Specifies the directory for the data files. Default = **.** See "Managing the Call Assembling Data Files".	No
RejectMissingChain	Specifies whether to report an error if a chain reference value is missing. Default = **False**	No
SplitAtGaps	Specifies whether a non-contiguous set of CDRs can be collected into a single EDR. For example, assume that CDRs F, I1, I2, and I4 have arrived. If set to **True**, this entry directs FCT_CallAssembling to emit an EDR for F, I1, and I2, and because I3 is missing, a separate EDR for I4. If set to **False**, all CDRs that have arrived are collected into a single EDR. If set to **True**, FCT_CallAssembling will emit multiple EDRs if a CDR is missing when the call. **True** = Active **False** = Inactive Default = **False** See "Rating Calls by Volume of Data Sent ".	No

Startup Registry Interdependencies

The Following Section Explains The Relationships Between Certain Startup Registry Entries.

Sample Registry

```
CallAssembling
{
   ModuleName = FCT_CallAssembling
   Module
    {
      Active = True
      Path = .
      FileName = calls
      RejectMissingChain = False
      AssembleVolume = TRUE
      AssembledEDR  {
          1 = Detail.custom_fields_from_last_edr1
          2 = Detail.custom_field_from_last_edr2...
       }
     }
}
```

Semaphore File Entries

Table 81-25 lists the FCT_CallAssembling Semaphore file entries.

Table 81-25 FCT_CallAssembling Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
ExportDataToXml	Exports the call data in the existing data file to an XML file with the name specified by the **FileName** entry in the startup registry file. See "Migrating Call Assembling Data Between Releases and Pipelines".
ExportDataToXML.CallsPerFile	If the number of calls exported is larger than the resources available in the host system, you can divide the call data into multiple files by using this option and specifying the number of calls per file. See "Migrating Call Assembling Data Between Releases and Pipelines".
FlushLimit	Sets the maximum age (in days) an open (incomplete) EDR can have before being flushed from the work files. For example, a setting of **0** flushes all open calls; a setting of **1** flushes all calls that have been open for a day or more; a setting of **2** flushes all calls that have been open for two days or more, and so on. **Note:** The setting of **0** does not flush future-dated EDRs because the value of CHARGING_START_TIMESTAMP is greater than the system date. No default value. See "Rating Incomplete Time Duration Calls ".
FlushServiceCode	Used with **FlushLimit**. Specifies a service. When used, only the calls with the service that match the three-letter service code are flushed. Multiple entries are not allowed. No default value. See "Rating Partial Calls by Service".
ImportDataFromXml	Imports the entire contents of the XML file created by the **ExportDataToXml** entry to the **.dat** file in the new format. **Values:** See "Migrating Call Assembling Data Between Releases and Pipelines".
ImportDataFromXML.FileName	Specifies the XML file from which to import data. See "Migrating Call Assembling Data Between Releases and Pipelines".
KeepCallOpen	Used with **FlushLimit**. Specifies whether to rate additional EDRs for a call that has already been flushed (**True**). Default = **False** See: • Rating Calls by Implied Time Duration • Rating Continuous Data Calls by Segment • Rating Partial Calls by Service
RemoveLimit	Sets a time limit (in days) for removing EDRs in a **Closed** or **Timeout** state from the work files. No default value. See "Removing Incomplete Time Duration Calls".
RemoveRejectedLimit	Sets a time limit (in days) for removing EDRs in a **Closed_Rejected** or **Timeout_Rejected** state from the work files. No default value. See "Removing Incomplete Time Duration Calls".

Table 81-25 (Cont.) FCT_CallAssembling Semaphore File Entries

Entry	Description
UpgradeFlushLimit	Flushes partial EDRs that were closed as a result of a change to the EDR container. No default value (no limit).

Sample FlushLimit Semaphore Commands

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.Module.FlushLimit=1

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.
Module.FlushServiceCode = tel

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.
Module.KeepCallOpen = True
```

Semaphore Entries for a Call-Assembling Report

For information, see "Tracking the Status of Assembled Calls".

Table 81-26 lists the semaphore entries for Call-Assembling Report.

Table 81-26 Semaphore Entries for Call-Assembling Report

Entry	Description	Mandatory
CreateReport	Command to create the report.	Yes
EndTime	Specifies the end date and time for the report. EDRs created before this date and time are reported. The format is *YYYYMMDDhhmmss*. Default = **0** (Current time)	No
ReportPath	Specifies path of the report file. Default = **.**	No
ReportPrefix	Specifies the file name prefix of the report file. Default = **assembly**	No
StartTime	Specifies the start date and time for the report. EDRs created on or after this date and time are reported. The format is *YYYYMMDDhhmmss*.	No

Sample semaphore command for call assembling reports

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CallAssembling.Module
{
     CreateReport      = True
     StartTime         = 20111206000000
     Module.StartTime  = 20020315000000
     EndTime           = 20111212090000
     ReportPath        = ./
     ReportPrefix      = call_assembly
}
```

EDR Container Fields

Table 81-27 lists the EDR container fields for Call-Assembling Report.

Table 81-27 EDR Container Fields for Call-Assembling Report

Alias Field Name Default Field Name	Type	Access	Description
CHAIN_REFERENCE DETAIL.CHAIN_REFERENCE	String	Read	Contains the chain reference key.
LONG_DURATION_INDICATOR DETAIL.LONG_DURATION_INDICATOR	String	Read/Write	Contains the long duration indicator. Arriving call segments have one of these: • **F** = First • **I** = Intermediate • **L** = Last Assembled call segments are given one of these: • **C** = Complete call. • **SL** = *Slice* (portion) of a call. • **P** = Partially assembled call. • **XC** = Late intermediate call segment. • **XO** = Late overlap segment. • **XP** =Late segment (any) of a call. See "How FCT_CallAssembling Classifies EDRs".
TRANSACTION_ID	Decimal	Read	Contains the transaction ID.
PROCESS_STATUS	Long	Read	Contains the EDR status.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read/Write	Contains the charging time stamp.
DURATION DETAIL.DURATION	Decimal	Read/Write	Contains the duration of the assembled EDR.
VOLUME_SENT DETAIL.VOLUME_SENT	Decimal	Read/Write	Contains the volume sent for the assembled EDR.
VOLUME_RECEIVED DETAIL.VOLUME_RECEIVED	Decimal	Read/Write	Contains the volume received for the assembled EDR.
NUMBER_OF_UNITS DETAIL.NUMBER_OF_UNITS	Decimal	Read/Write	Contains the number of units for the assembled EDR.
RETAIL_CHARGED_AMOUNT_VALUE DETAIL.RETAIL_CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the retail charged amount value for the assembled EDR.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the wholesale charged amount value for the assembled EDR.

Table 81-27 (Cont.) EDR Container Fields for Call-Assembling Report

Alias Field Name Default Field Name	Type	Access	Description
NUMBER_OF_CDRS DETAIL.NUMBER_OF_CDRS	Integer	Write	The number of CDRs assembled in the EDR.

FCT_CancelTimer

The FCT_CancelTimer module checks the TimerID to identify the EDR and the timeout flag to verify if the EDR is valid or timed out. If the time out flag is set to **False**, FCT_CancelTimer cancels the timeout flag in the EDR so that the EDR can be sent for further processing.

If the timeout flag is set to **True**, it means there is a duplicate EDR and the FCT_CancelTimer discards the EDR.

Dependencies

FCT_CancelTimer depends on the FCT_Timer in the Dispatcher pipeline for the TimerID and the timeout flag values.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-28 lists the FCT_CancelTimer registry entries.

Table 81-28 FCT_CancelTimer Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
StreamName	The output queue to which the timed out EDR is sent.	Yes

Sample Registry

```
CancelTimer
{
  ModuleName = FCT_CancelTimer
  Module
  {
    Active = TRUE
    StreamName = ExceptionOutput
  }
}
```

EDR Container Fields

FCT_CancelTimer uses the EDR container fields listed in Table 81-29:

Table 81-29 FCT_CancelTimer Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TIMER_ID DETAIL.TIMER_ID	Integer	Read	Contains the timer ID needed to cancel the timer

FCT_CarrierIcRating

The FCT_CarrierIcRating module adds roaming/interconnect data to EDRs for rating by the FCT_PreRating and FCT_MainRating modules.

Dependencies

Requires a connection to the Pipeline Manager database.

All rating and mapping related modules should be placed in the pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-30 lists the FCT_CarrierIcRating registry entries.

Table 81-30 FCT_CarrierIcRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
EdrNetworkModel	Specifies the network model. This entry identifies your network as the home network. You can specify one home network per pipeline. You can use this entry in a semaphore file.	Yes
IcProductGroup	Specifies the IC charge offer group that contains the IC charge offers. This field is mandatory for all modes except CARRIER_IC mode.	Yes
InterConnectDataModule	Specifies a connection to the DAT_Interconnect module.	Yes
Mode	Specifies the evaluation path for finding the IC-charge offer. • If you specify ROAMING, IC charge offers are found using the **IcProductGroup** registry entry. • If you specify CARRIER_IC, the module assigns a charge by using trunk information from the EDR.	Yes

Table 81-30 (Cont.) FCT_CarrierIcRating Registry Entries

Entry	Description	Mandatory
RecordTypeField	Specifies the EDR field that contains the record type. The record type is used to search the IFW_ICPRODUCT_CNF database table for matching records. When processing CIBER record types, this entry is used to find the IC Charge Offers and the corresponding charge to use for rating the CIBER records.	No
UseRateplan	Specifies how the price is calculated: • **STANDARD**. The price is calculated using the specified charge. • **ALTERNATIVE**. The price is calculated using the alternative charge. If you entered an alternative charge when configuring the IC charge offer, you can specify whether to use the alternate charge. You can use this entry in a semaphore file.	Yes

Sample Registry

```
Module
{
  Active = TRUE
  InterConnectDataModule = integRate.DataPool.InterConnect
  EdrNetworkModel = OWN
  UseRateplan = STANDARD
  Mode = ROAMING
  IcProductGroup = PG_ROAM
  RecordTypeField = DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE
}
```

Semaphore File Entries

Table 81-31 lists the FCT_CarrierIcRating Semaphore file entries.

Table 81-31 FCT_CarrierIcRating Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrNetworkModel	Specifies the network model to be used (CODE from table IFW_NETWORKMODEL).
UseRateplan	• **STANDARD:** IC-Price will be calculated using the charge from IFW_ICPRODUCT_RATE. • **ALTERNATIVE:** IC-Price will be calculated using the alternative charge from IFW_ICPRODUCT_RATE.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CarrierIcRating.Module.Active = TRUE

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
```

```
CarrierIcRating.Module.EdrNetworkModel = OTHER

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CarrierIcRating.Module.UseRateplan = ALTERNATIVE
```

EDR Container Fields

Table 81-32 lists the FCT_CarrierIcRating EDR container fields.

Table 81-32 FCT_CarrierIcRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Create	The associated charge breakdown record created to hold the mapping data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	Create	The charge packet created to hold the mapping data.
TRUNK_INPUT DETAIL.ASS_GSMW_EXT.TRUNK_INPUT	String	Read	Contains the input trunk search value from the IFW_TRUNK_CNF table.
TRUNK_OUTPUT DETAIL.ASS_GSMW_EXT.TRUNK_OUTPUT	String	Read	Contains the output trunk search value from the IFW_TRUNK_CNF table.
ASS_GSMW_ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the switch search value from the IFW_TRUNK_CNF table.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the IC charge offer search value from the IFW_ICPRODUCT_CNF table.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network search value from the IFW_ICPRODUCT_CNF table.
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the A number search value from the IFW_ICPRODUCT_CNF table.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number search value from the IFW_ICPRODUCT_CNF table.
C_NUMBER DETAIL.C_NUMBER	String	Read	Contains the C number search value from the IFW_ICPRODUCT_CNF table.
RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type search value from the IFW_ICPRODUCT_CNF table.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code search value from the IFW_ICPRODUCT_CNF table.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class search value from the IFW_ICPRODUCT_CNF table.

Table 81-32 (Cont.) FCT_CarrierIcRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class search value from the IFW_ICPRODUCT_CNF table.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the internal A number zone. This value sets the charge packet INTERN_ORIGIN_NUM_ZONE value.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the internal B number zone. This value sets the charge packet INTERN_DESTIN_NUM_ZONE value.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type: • **990** = CarrierIC • **991** = Roaming
INTERN_CALC_MODE DETAIL.ASS_CBD.INTERN_CALC_MODE	String	Write	Contains the calculation mode from the CALCMODE field in the IFW_NETWORKMODEL database table.
CHARGE_TYPE DETAIL.ASS_CBD.CP.CHARGE_TYPE	String	Write	Contains the charge type.
NETWORK_OPERATOR DETAIL.ASS_CBD.CP.NETWORK_OPERATOR_CODE	String	Write	Contains the network operator code from the CONNECTED_NO field in the IFW_TRUNK database table.
NETWORK_OPERATOR_BILLINGTYPE DETAIL.ASS_CBD.CP.NETWORK_OPERATOR_BILLINGTYPE	String	Write	Contains the billing type from the BILL_DIRECTION field in the IFW_ICPRODUCT_RATE database table.
PRODUCTCODE_USED DETAIL.ASS_CBD.CP.PRODUCTCODE_USED	String	Write	Contains the IC charge offer code from the ICPRODUCT field in the IFW_ICPRODUCT database table.
TRUNK_USED DETAIL.ASS_CBD.CP.TRUNK_USED	String	Write	Contains the trunk from the TRUNK field in the IFW_TRUNK database table.
POI_USED DETAIL.ASS_CBD.CP.POI_USED	String	Write	Contains the POI from the POI field in the IFW_POI database table.
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging time stamp.
INTERN_FIX_COST DETAIL.ASS_CBD.CP.INTERN_FIX_COST	Decimal	Write	Contains the internal fixed cost. Added to the charge by the FCT_MainRating module.

Table 81-32 (Cont.) FCT_CarrierIcRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Write	Contains the charge. Used by the FCT_PreRating and FCT_MainRating modules.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Write	Contains the A number zone. Used by the FCT_PreRating module to find the impact category.
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Write	Contains the B number zone. Used by the FCT_PreRating module to find the impact category.
INTERN_BILLING_CURRENCY DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY	String	Write	Contains the billing currency name
INTERN_HOME_CURRENCY DETAIL.ASS_CBD.CP.INTERN_HOME_CURRENCY	String	Write	Contains the home currency name

FCT_CiberOcc

The FCT_CiberOcc module creates a CIBER record for other charges and credits (OCC record), type 50 or 52.

Dependencies

This module requires a connection to the DAT_InterConnect module.

Must run after the FCT_DuplicateCheck module and before the FCT_CarrierIcRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-33 lists the FCT_CiberOcc registry entries.

Table 81-33 FCT_CiberOcc Registry Entries

Entry	Description	Mandatory	Default
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes	N/A

Table 81-33 (Cont.) FCT_CiberOcc Registry Entries

Entry	Description	Mandatory	Default
CallRecordTypeField	Specifies the EDR field that indicates the CIBER record type. When processing CIBER record types, this entry is used to find the IC Charge Offers and the corresponding charge to use for rating the CIBER records.	No	DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE
EdrNetworkModel	Specifies the network model to use for CIBER_OCC searching. This identifies the home network You can specify one home network per pipeline. You can change this value by using a semaphore.	Yes	N/A
InterConnectDataModule	Specifies a connection to the DAT_InterConnect module.	Yes	N/A
NoOCCField	Specifies the field that indicates whether to generate an OCC record. This field must match the field name specified in the **DuplicateIndicatorField** entry of the FCT_DuplicateCheck registry. See "FCT_DuplicateCheck".	Yes	DETAIL.ASS_CIBER_EXT.NO_OCC
OCCDescription	Description of the service associated with the OCC. **Important:** This field must not contain spaces. If you require spaces in the description, write an iScript to populate this field.	No	" " (Empty string)
OCCIntervalIndicator	Specifies the interval at which the associated OCC record is generated.	No	**3**

Sample Registry

```
CiberOcc
{
   ModuleName = FCT_CiberOcc
   Module
   }
      Active = TRUE
      InterConnectDataModule = ifw.DataPool.InterConnect
      EdrNetworkModel = ROAMING
      CallRecordTypeField = DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE
      NoOCCField = DETAIL.ASS_CIBER_EXT.NO_OCC
      OCCIntervalIndicator = 3
      OCCDescription = DAILY_SURCHARGE
    }
}
```

Semaphore File Entries

Table 81-34 lists the FCT_CiberOcc Semaphore file entries.

Table 81-34 FCT_CiberOcc Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrNetworkModel	The network model to use for CIBER_OCC searching.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CiberOcc.Module.Active = false

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.
CiberOcc.Module.EdrNetworkModel = ROAMING
```

EDR Container Fields

The FCT_CiberOcc module uses the EDR container fields listed in Table 81-35:

Table 81-35 FCT_CiberOcc EDR Container Fields

Default Field Name	Type	Access	Description
DETAIL.ASS_CIBER_EXT.AIR_CONNECT_TIME	Date	Read	Used to specify the connection time for the OCC record.
DETAIL.CHARGING_START_TIMESTAMP	Date	Read	IFW_CIBER_OCC
DETAIL.ASS_CIBER_EXT.CHARGE_NO_1_CONNECT_TIME	Date	Read	Used to specify the connection time for the OCC record.
DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE	String	Read	This field is specified in the **CallRecordTypeField** entry in the registry. This is the default field used to determine the current EDR call record type.
DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE	String	Write	Specifies the type of record to create. If the current record type is: • 22 or 32, assign **52** to this field. • 10, 20, or 30, assign **50** to this field.
DETAIL.ASS_CIBER_EXT.CONNECT_TIME	Date	Write	This field is set to one of the following: • **AIR_CONNECT_TIME** if the call record type is 22. • **SSU_CONNECT_TIME** if the call record type is 10 or 20. • **CHARGE_NO_1_CONNECT_TIME** if the call record type is 30 or 32.

Table 81-35 (Cont.) FCT_CiberOcc EDR Container Fields

Default Field Name	Type	Access	Description
DETAIL.ASS_CIBER_EXT.NO_OCC	String	Read	This field is specified in the **NoOCCField** entry in the registry.
DETAIL.ASS_CIBER_EXT.OCC_DESCRIPTION	String	Write	The value of this field is specified in the **OCCDescription** entry in the registry. The default value is an empty string: " "
DETAIL.ASS_CIBER_EXT.OCC_END_DATE	Date	Write	The value of **CHARGING_START_TIMESTAMP** in the current EDR.
DETAIL.ASS_CIBER_EXT.OCC_INTERVAL_INDICATOR	String	Write	The value of this field is specified in the **OCCIntervalIndicator** entry in the registry. The default value is **3** (daily interval).
DETAIL.ASS_CIBER_EXT.OCC_START_DATE	Date	Write	The value of **CHARGING_START_TIMESTAMP** in the current EDR.
DETAIL.ASS_CIBER_EXT.RECORD_CREATE_DATE	Date	Write	This field is set to the system date.
DETAIL.ASS_CIBER_EXT.RECORD_USE_INDICATOR	String	Write	This field is set to **1**.
DETAIL.ASS_CIBER_EXT.SEQ_INDICATOR	String	Write	This field is set to **01**.
DETAIL.SOURCE_NETWORK	String	Read	Search value used for searching the IFW_CIBER_OCC database table.
DETAIL.ASS_CIBER_EXT.SSU_CONNECT_TIME	Date	Read	Used to specify the connection time for the OCC record.

Database Interface for the FCT_CiberOcc Module

The FCT_CiberOcc module uses the IFW_CIBER_OCC database table to determine whether OCC records are generated for the network operator.

FCT_CliMapping

The FCT_CliMapping module maps multiple numbers to a single number for billing. See "Mapping Multiple Phone Numbers to a Single Number".

Dependencies

Must run before the rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-36 lists the FCT_CliMapping registry entries.

Table 81-36 FCT_CliMapping Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
MapFile	Specifies the path to the mapping file. You can use this entry in a semaphore file.	Yes

Sample Registry Entry

```
CliMapping
{
  ModuleName = FCT_CliMapping
  Module
   {
    Active = True
    MapFile = cli_map_1.dat
  }
}
```

Semaphore File Entries

Table 81-37 lists the FCT_CliMapping Semaphore file entries.

Table 81-37 FCT_CliMapping Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FucntionPool.CliMapping.Module.Active = True
```

EDR Container Fields

Table 81-38 lists the FCT_CliMapping EDR container fields.

Table 81-38 FCT_CliMapping EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read/Write	Contains the customer A number.
CUST_A_ACCOUNT_ID DETAIL.CUST_A.ACCOUNT_ID	Block	Read	Contains the customer account ID.

FCT_CreditLimitCheck

The FCT_CreditLimitCheck module determines whether event owners have enough balances in their account balance to cover the cost of usage. If the account does not have sufficient balances to authorize the entire request, this module determines how much usage can be authorized with the available balances.

> **Note:**
>
> The FCT_CreditLimitCheck module does not check the credit floor.

Dependencies

Use this module in a real-time discounting pipeline.

This module must run after all other discounting modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-39 lists the FCT_CreditLimitCheck registry entries.

Table 81-39 FCT_CreditLimitCheck Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
CLCTrace	Specifies whether to generate a credit limit check trace file. **True** = Generate a trace file. **False** = Do not generate a trace file (Default)	No
CurrencyDataModule	Specifies the connection to the DAT_Currency module.	Yes
RoundUpRequestQuantity	Determines whether authorized quantities are rounded up to remove fractional values. **True** = Round up **False** = No rounding (Default)	No
StepValue	Specifies the step value to be considered for the quantity during reverse rating and for rounding the prorated quantity.	No

Sample Registry Entry

```
#------------------------------------------------
# Credit Limit Check
#------------------------------------------------
CreditLimitCheckModule
{
  ModuleName = FCT_CreditLimitCheck
```

```
  Module
  {
    Active  = True
    RoundUpRequestQuantity = True
    CLCTrace = True
    CurrencyDataModule = ifw.DataPool.CurrencyDataModule
    StepValue = 0.1
  }
}
```

EDR Container Fields

The FCT_CreditLimitCheck module uses the EDR container fields listed in Table 81-40:

Table 81-40 FCT_CreditLimitCheck EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CREDIT_LIMIT_CHECK DETAIL.CREDIT_LIMIT_CHECK	Integer	Read	Specifies whether to perform a credit limit check on the EDR: 1 = check; 0 = don't check.
CREDIT_LIMIT_CHECK_RESULT DETAIL.CREDIT_LIMIT_CHECK_RESULT	Integer	Write	Specifies whether the credit limit check passed or failed: 1 = passed; 0 = failed.
INTERN_BALANCE_GROUP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	Read	Account level balance group of the event owner account.
DETAIL.UNRATED_QUANTITY	Decimal	Write	The quantity that could not be rated.
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Read	Balance Element ID for the charge packet.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read	The balance impact of the charge packet. This amount was computed by real-time rating.
QUANTITY_FROM DETAIL.ASS_CBD.CP.QUANTITY_FROM	Decimal	Read	Charge packet start quantity. If the charge packet is split by FCT_Discount, this module reads QUANTITY_FROM values from the DETAIL.ASS_CBD.CP.SPLIT_CP block.

Table 81-40 (Cont.) FCT_CreditLimitCheck EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
QUANTITY_TO DETAIL.ASS_CBD.CP.QUANTITY_TO	Decimal	Read	Charge packet end quantity. If the charge packet is split by FCT_Discount, this module reads QUANTITY_TO values from the DETAIL.ASS_CBD.CP.SPLIT_CP block.
DETAIL.ASS_CBD.CP.ROUNDED_QTY_VALUE	Decimal	Read	The quantity that could be authorized.
DP_DISCOUNT_BALANCE_GROUP_ID DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	Integer	Read	POID of the balance group impacted by this discount packet.
DP_DISCOUNT_GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read	Total amount of the discount packet. This discount amount is applied to the balance group.
DP_QUANTITY_FROM DETAIL.ASS_CBD.DP.QUANTITY_FROM	Decimal	Write	Discount packet start quantity. Aligns with the QUANTITY_FROM value in a charge packet or a split charge packet.
DP_QUANTITY_TO DETAIL.ASS_CBD.DP.QUANTITY_TO	Decimal	Write	Discount packet end quantity. Aligns with the QUANTITY_TO value in a charge packet or a split charge packet.
DP_DISCOUNT_RESOURCE_ID DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Read	Balance Element ID for the discount packet.
BG_BG_ID DETAIL.CUST_A.BG.BALANCE_GROUP_ID	String	Read	Numeric ID for the balance group.
BG_BELEM_RESOURCE_ID DETAIL.CUST_A.BG.BAL_ELEM.RESOURCE_ID	Integer	Read	The balance element ID for the balance group element.
BG_BELEM_CURR_BAL DETAIL.CUST_A.BG.BAL_ELEM.CURR_BAL	Decimal	Read	The event owner's current balance for this balance group element.
BG_BELEM_CREDIT_LIMIT DETAIL.CUST_A.BG.BAL_ELEM.CREDIT_LIMIT	Decimal	Read	The credit limit for this balance group element.
BG_BELEM_RESERVED_AMOUNT DETAIL.CUST_A.BG.BAL_ELEM.RESERVED_AMOUNT	Decimal	Read	The amount already in reserve by the event owner.
DETAIL.RUM_MAP.RUM_NAME	String	Read	Name of the RUM. Used in multi-RUM checks.
DETAIL.RUM_MAP.NET_QUANTITY	Decimal	Write	The total requested quantity for a RUM. Used in multi-RUM checks.

Table 81-40 (Cont.) FCT_CreditLimitCheck EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.RUM_MAP.UNRATED_QUANTITY	Decimal	Write	The quantity of a RUM that could not be authorized. Used in multi-RUM checks.

FCT_CustomerRating

The FCT_CustomerRating module supplies charges for the FCT_MainRating module.

See the following documents:

- About Customer Rating
- FCT_MainRating

The FCT_CustomerRating module is also used for least-cost rating and promotional overlays. See:

- About Least Cost Rating
- About Calculating the Promotional Savings

Dependencies

Requires a connection to the Pipeline Manager database.

This module must run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-41 lists the FCT_CustomerRating registry entries.

Table 81-41 FCT_CustomerRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
DefaultRateplan	Specifies the charge code used as default if no customer data for the A number can be found. You can use this entry in a semaphore file. See "Assigning a Default Charge and Default Segment for Customer Rating".	No

Table 81-41 (Cont.) FCT_CustomerRating Registry Entries

Entry	Description	Mandatory
DefaultSegment	Specifies the segment name used as default if no customer data for the A number can be found. You can use this entry in a semaphore file. See "Assigning a Default Charge and Default Segment for Customer Rating".	No
LeastCostRating	Specifies whether least cost rating is active or inactive. For more information, see "About Least Cost Rating". • **True** activates least cost rating. • **False** disables least cost rating. See "Configuring Least Cost Rating".	No
Mode	Specifies that the module is run for customer rating (**CUSTOMER**). See "About Customer Rating".	Yes
PromotionalSaving	Specifies whether to calculate the total savings to customers when rating a usage event with a promotional charge offer rather than a base charge offer. For more information, see "About Calculating the Promotional Savings". • **True** specifies to calculate the savings amount. • **False** specifies to not calculate the savings amount. See "About Calculating the Promotional Savings".	No

Sample Registry

```
CustomerRating
{
  ModuleName = FCT_CustomerRating
  Module
  {
    Active = True
    Mode = CUSTOMER
    DataConnection = ifw.DataPool.Database
  }
}
```

Semaphore File Entries

Table 81-42 lists the FCT_CustomerRating Semaphore file entries.

Table 81-42 FCT_CustomerRating Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
DefaultRateplan	Specifies the charge code used as default if no customer data for the A number can be found. See "Assigning a Default Charge and Default Segment for Customer Rating".

Table 81-42 (Cont.) FCT_CustomerRating Semaphore File Entries

Entry	Description
DefaultSegment	Specifies the segment name used as default if no customer data for the A number can be found. See "Assigning a Default Charge and Default Segment for Customer Rating".

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.CustomerRating.Module.Active = False
```

EDR Container Fields

Table 81-43 lists the FCT_CustomerRating EDR container fields.

Table 81-43 FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Create	The associated charge breakdown record created to hold the rating data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD_CHARGE_PACKET	Block	Create	The charge packet created to hold the rating data.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. Written to the DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP field.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the A number zone. Written to the DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE field.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number zone. Written to the DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE field.
INTERN_SLA_RSC_GROUP DETAIL.INTERN_SLA_RSC_GROUP	String	Write	Contains the SLA RSC group.
INTERN_SLA_USC_GROUP DETAIL.INTERN_SLA_USC_GROUP	String	Write	Contains the SLA USC group code.
INTERN_SLA_IRULE_SET DETAIL.INTERN_SLA_IRULE_SET	String	Write	Contains the SLA iRule set.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type for the associated charge breakdown record. • 981 = Customer rating

Table 81-43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGE_TYPE DETAIL.ASS_CBD.CP.CHARGE_TYPE	String	Write	Contains the charge type. This field is always set to **N**.
ASS_CBD_ACCOUNT_CODE DETAIL.ASS_CBD.ACCOUNT_CODE	String	Write	Contains the account code. Set with the value from the DETAIL.CUST_A.ACCOUNT_NO field.
ASS_CBD_SYSTEM_BRAND_CODE DETAIL.ASS_CBD.SYSTEM_BRAND_CODE	String	Write	Contains the system brand code. Set with the value from the DETAIL.CUST_A.SYSTEM_BRAND field.
ASS_CBD_CUSTOMER_BILLCYCLE DETAIL.ASS_CBD.CUSTOMER_BILLCYCLE	String	Write	Contains the customer's bill cycle code. Set with data from the DETAIL.CUST_A.BILL_CYCLE field.
ASS_CBD_CUSTOMER_CURRENCY DETAIL.ASS_CBD.CUSTOMER_CURRENCY	String	Write	Contains the customer's currency. Set with the value from the DETAIL.CUST_A.CURRENCY field.
ASS_CBD_CUSTOMER_RATEPLAN_CODE DETAIL.ASS_CBD.CUSTOMER_RATEPLAN_CODE	String	Write	A comma-separated list of charge codes for all rating charge offers. The list is arranged by charge offer priority, with highest priority first and lowest priority last.
INTERN_BILLING_CURRENCY DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY	String	Write	Contains the billing currency. Set with the value from the DETAIL.CUST_A.CURRENCY field.
ASS_CBD_SEGMENT_CODE DETAIL.ASS_CBD.SEGMENT_CODE	String	Write	Contains the segment code. Set with the value from the Data Warehouse ERA.
ASS_CBD_SLA_CODE DETAIL.ASS_CBD.SLA_CODE	String	Write	Contains the SLA code. Set with the value from the Service Level Agreement ERA.
INTERN_DISCOUNT_ACCOUNT DETAIL.ASS_CBD.CP.INTERN_DISCOUNT_ACCOUNT	String	Write	Contains the discount account. Set with the value from the Discount ERA.

Table 81-43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	Contains the charge code to use for rating.
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging time stamp. Set with the value from the DETAIL.CHARGING_START_TIMESTAMP field.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Write	Contains the internal charge code. Set with values from the Data Warehouse ERA.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Write	Contains the zone for the A number. Set with DETAIL.INTERN_A_NUMBER_ZONE
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Write	Contains the zone for the B number. Set with DETAIL.INTERN_B_NUMBER_ZONE
CUST_A_ACCOUNT_ID DETAIL.CUST_A.ACCOUNT_ID	String	Read	Contains the customer account ID. Write to DETAIL.ASS_CBD.ACCOUNT_CODE
CUST_A_ACCOUNT_NO DETAIL.CUST_A.ACCOUNT_NO	String	Read	Contains the customer account number.
CUST_A_SYSTEM_BRAND DETAIL.CUST_A.SYSTEM_BRAND	String	Read	Contains the system brand. Written to the DETAIL.ASS_CBD.SYSTEM_BRAND_CODE field.
CUST_A_BILL_CYCLE DETAIL.CUST_A.BILL_CYCLE	String	Read	Contains the customer's bill cycle. Written to the DETAIL.ASS_CBD.CUSTOMER_BILLCYCLE field.
CUST_A_CURRENCY DETAIL.CUST_A.CURRENCY	String	Read	Contains the customer's currency. Written to the DETAIL.ASS_CBD.CUSTOMER_CURRENCY and DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY field.

Table 81-43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CUST_A_INTERN_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Contains an index of the customer's purchased charge offers identified by the FCT_Account module.
CUST_A_RATEPLAN_NAME DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Read	Contains the charge name. Written to the DETAIL.ASS_CBD.CP.RATEPLAN_CODE field.
CUST_A_PROFILE DETAIL.CUST_A.ERA.PROFILE	String	Read	Contains the customer's account-related ERA data.
CUST_A_KEY DETAIL.CUST_A.ERA.PA.KEY	String	Read	Contains the key for the account-related ERA data.
CUST_A_VALUE DETAIL.CUST_A.ERA.PA.VALUE	String	Read	Contains the value for the account-related ERA data.
CUST_A_PRODUCT_PROFILE DETAIL.CUST_A.PRODUCT.ERA.PROFILE	String	Read	Contains the customer's service-related ERA data.
CUST_A_PRODUCT_KEY DETAIL.CUST_A.PRODUCT.ERA.PA.KEY	String	Read	Contains the key for the service-related ERA data.
CUST_A_PRODUCT_KEY DETAIL.CUST_A.PRODUCT.ERA.PA.KEY	String	Read	Contains the value for the service-related ERA data.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Read	Contains the charge offer rating indexes. This is a comma-separated list of all rating charge offers' indexes associated with the same service and event, and their priorities.
DETAIL.CUST_A.LEAST_COST	Integer	Write	Indicates whether to use least cost rating for an EDR. **1** turns it off; any other integer turns it on. This entry overrides the least cost rating entry in the registry file.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all charge offers that are associated with the same service and event.
DETAIL.CUST_A.PRODUCT.USAGE_START	Date	Read	Contains a list of the start times for all charge offers that are associated with the same service and event.

Table 81-43 (Cont.) FCT_CustomerRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.CUST_A.PROMOTIONAL_SAVING	Integer	Write	Indicates whether to calculate the promotional savings for an EDR. **1** turns off promotional savings; any other integer turns it on.
DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	A comma-separated list of charge codes for all rating charge offers. The list is arranged by charge offer priority, with highest priority first and lowest priority last.

FCT_Dayrate

The FCT_Dayrate module calculates charges for special day rates, for example, a discount for calls made on January 1.

Dependencies

Requires a connection to the DAT_Dayrate module.

This module must run after the FCT_MainRating module to adjust the rate.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-44 lists the FCT_Dayrate registry entries.

Table 81-44 FCT_Dayrate Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	No
DayrateDataModule	Specifies the connection to the DAT_Dayrate module.	No

Sample Registry

```
Dayrate
{
  ModuleName = FCT_Dayrate
  Module
  {
    Active = True
    DayrateDataModule = DayrateData
```

```
    }
}
```

Semaphore File Entries

Table 81-45 lists the FCT_Dayrate Semaphore file entry.

Table 81-45 FCT_Dayrate Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SpecialDayRate.Module.Active = False
```

EDR Container Fields

Table 81-46 lists the FCT_Dayrate EDR container fields.

Table 81-46 FCT_Dayrate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. The field is used to determine which discount to use.
ASS_CBD.CP.INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the charge code. This field is used to determine which discount to use.
ASS_CBD.CP.INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Block	Read	Contains the charge version. This field is used to determine which discount to use.
ASS_CBD.CP.CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Block	Read/Write	Contains the recalculated charged amount value.
ASS_CBD.CP DETAIL.ASS_CBD.CP	Block	Read	Contains the block index to charge packages.

FCT_Discard

The FCT_Discard module discards or skips EDRs based on configurable EDR properties.

- Skipping an EDR removes it from the pipeline.
- Discarding an EDR sends it to a different output stream.

In both the cases the state of the EDR becomes invalid.

See "Discarding and Skipping EDRs ".

Dependencies

Requires a connection to the Pipeline Manager database.

Because you can discard or split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap module. Should be early in the function pool, but must be run after FCT_CallAssembling.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-47 lists the FCT_Discard registry entries.

Table 81-47 FCT_Discard Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
Function	Specifies whether to discard or skip the EDR. Default = **Discard** You can use this entry in a semaphore file.	No
StreamName	Specifies the output stream for discarded EDRs. You can use this entry in a semaphore file. Default = **DevNull** See "Configuring Output of Discarded EDRs".	No

Sample Registry

```
Discard
{
   ModuleName = FCT_Discard
   Module
   {
      Active = True

      DataConnection = ifw.DataPool.Database
      Function = Discard
      StreamName = DevNull
   }
}
```

Semaphore File Entries

Table 81-48 lists the FCT_Discard Semaphore file entries.

Table 81-48 FCT_Discard Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Function	Specifies whether to discard or skip the EDR. Default = **Discard**
Reload	Reloads the discard rules.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.EventDiscarding.Module.Reload {}
```

EDR Container Fields

Table 81-49 lists the FCT_Discard EDR container fields.

Table 81-49 FCT_Discard EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network code. This could either be PLMN ID or any logical operator code.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network code.
CALL_COMPLETION_INDICATOR DETAIL.CALL_COMPLETION_INDICATOR	String	Read	Indicates if a call was successfully completed.
LONG_DURATION_INDICATOR DETAIL.LONG_DURATION_INDICATOR	String	Read	Contains the long duration indicator: • **F** = First • **I** = Intermediate • **L** = Last
USAGE_CLASS DETAIL.USAGE_CLASS	String	Read	Contains the external usage class.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GSM MSC or Switch ID handling the origin of the call.
ASS_GPRS_EXT_ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GPRS MSC or Switch ID handling the origin of the call.

Table 81-49 (Cont.) FCT_Discard EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TARIFF_CLASS DETAIL.TARIFF_CLASS	String	Read	Contains the tariff class.
TARIFF_SUB_CLASS DETAIL.TARIFF_SUB_CLASS	String	Read	Contains the tariff subclass.
CONNECT_SUB_TYPE DETAIL.CONNECT_SUB_TYPE	String	Read	Contains the connection subtype.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Read	Contains the sum of the wholesale charged amount value.
DISCARDING DETAIL.DISCARDING	Integer	Write	Indicates if the EDR should be discarded.
DETAIL.DISCARD_REASON	String	Write	Specifies the name of the discarding rule that applies to the EDR. Discarding rules are defined in the ifw_discarding table. If any of the rules in the table applies to the EDR, the EDR is discarded or skipped.

Database Tables

The FCT_Discard module uses the data in the IFW_Discarding table to determine which EDRs should be discarded. See "Discarding and Skipping EDRs ".

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Discount

The FCT_Discount module performs discounting operations. The module supports discounting for events rated in real time and events rated in batch by Pipeline Manager.

When the module is called for discount calculation as part of prepaid authorization or reauthorization, it splits charge packets if necessary to create linear segments to which a single net rate applies.

Dependencies

Requires a connection to the following:

- The DAT_Discount module. See "DAT_Discount".
- A balance data module, either DAT_BalanceRealtime or DAT_BalanceBatch depending on whether the module is being used for real-time or batch discounting. See "DAT_BalanceRealtime" or "DAT_BalanceBatch".
- An account data module, either DAT_AccountRealtime or DAT_AccountBatch depending on whether the module is being used for real-time or batch discounting. See "DAT_AccountRealtime" or "DAT_AccountBatch".
- The DAT_Currency module. This module converts balance element codes to balance element IDs. See "DAT_Currency".

For batch discounting, you must also configure the FCT_ApplyBalance module, which should be in the same function pool as this module for performance reasons and must run after this module.

This module must run after FCT_MainRating.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-50 lists the FCT_Discount registry entries.

Table 81-50 FCT_Discount Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountRealtime or DAT_AccountBatch module.	Yes
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry as a semaphore command.	Yes
AvoidMatchFactorCalculation	Specifies whether to calculate the amount of usage that is discounted (the match factor) for parallel and sequential discounts. The match factor is typically used only for cascading discounts. See "Calculating the Match Factor of Parallel and Sequential Discounts".	No
BackOut	Specifies if the module is used in a back-out pipeline for rerating. Default = **False**	No
BalanceDataModule	Specifies the connection to the DAT_BalanceRealtime or DAT_BalanceBatch module.	Yes
CurrencyDataModule	Specifies the connection to DAT_Currency module.	Yes
DiscountDataModule	Specifies the connection to the DAT_Discount module.	Yes
DiscountMoreThanPossible	Specifies if a discount can be higher than the real revenue. Default = **False**	No

Table 81-50 (Cont.) FCT_Discount Registry Entries

Entry	Description	Mandatory
DiscountTrace	Specifies whether to generate discount trace file. • **True** indicates that a discount trace file is generated. • **False** indicates that a discount trace file is not generated. Default = **False**	No
EvalMultipleBalImpact	Specifies to create a discount balance impact for each charge packet when the discount base expression is not a simple expression (StepC, StepQ, TotalC, or TotalQ). Default = **False**	No
IgnoreEDROnDeadlock	Specifies whether to ignore EDRs causing the deadlock. • **True** indicates that the module should ignore the EDRs and continue processing the EDR file. The module places the EDR causing the deadlock into the discountError directory. • **False** indicates that the module should roll back already processed EDRs and start reprocessing the same file.	No
IgnoreEDROnDiscountError	Specifies whether to ignore EDRs with discounting error. • **True** indicates that the module should ignore the EDRs with discounting error and continue processing the remaining EDRs in the EDR file. The module places the EDRs that failed discounting into the discountError directory. • **False** indicates that the module should roll back the EDRs with discounting error that has already been processed and reprocess them. Default = **False**	No
IgnoreEDROnLock	Specifies whether to ignore the EDR if the balance group object is locked by another transaction. The ignored or rejected EDRs with the locked balance group are placed into the **discountError** directory. • **True** indicates that the module ignores the EDRs and continues processing the EDR file. • **False** indicates that the module rolls back already processed EDRs and begins reprocessing the same file. Default = **False**	No
ShowZeroDiscount	Specifies whether Discount Packet are generated when granted discount amount is 0. • **True** indicates that Discount Packets are generated when the granted discount amount is 0. • **False** indicates that Discount Packets are not generated when the granted discount amount is 0. Default = **False**	No
SupportBundleERA	Specifies if the support for ERAs is needed. Default = **False**	No

Table 81-50 (Cont.) FCT_Discount Registry Entries

Entry	Description	Mandatory
TaxationMode	Used only when FCT_Discount is configured for discounting in the real-time pipeline. Specifies when events are taxed. The value of this entry should be the same as the value of **taxation_switch** entry in the Connection Managers (CM) configuration file. Possible values are: **0** - Taxation is disabled. **1** - Enable real-time tax calculation. **2** - Enable deferred (cycle-time) tax calculation. **3** - Enable real-time and deferred tax calculation. Default = **3**	No
ZeroValuePacketFilterDisabled	Used with the aggregation scenario to filter out charge packets where either charge or quantity is zero. Default = **True**	No
ProrateFixedDiscount	Specifies whether to prorate fixed cycle-event discounts: • **False** indicates that fixed cycle-event discounts are not prorated. This is the default. • **True** indicates that fixed cycle-event discounts are prorated. To prorate fixed cycle-event discounts, you must set this entry to **True** before starting the real-time pipeline.	No

Sample Registry

```
GeneralDiscounting
{
  ModuleName = FCT_Discount
  Module
  {
    Active                    = TRUE
    DiscountDataModule        = ifw.DataPool.DiscountModelDataModule
    BalanceDataModule         = ifw.DataPool.BalanceDataModule
    AccountDataModule         = ifw.DataPool.CustomerData
    CurrencyDataModule        = ifw.DataPool.CurrencyDataModule
    DiscountMoreThanPossible  = False
    ProrateFixedDiscount      = True
    TaxationMode              = 3
    EvalMultipleBalImpact     = True
  }
}
```

Semaphore File Entries

Table 81-51 lists the FCT_Discount Semaphore file entries.

Table 81-51 FCT_Discount Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
IgnoreEDROnLock	Specifies whether to ignore the EDR if the balance group object is locked by another transaction. The ignored or rejected EDRs with the locked balance group are placed into the **discountError** directory. • **True** indicates that the module ignores the EDRs and continues processing the EDR file. • **False** indicates that the module rolls back already processed EDRs and begins reprocessing the same file. Default = **False**

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ApolloDiscountModule.
Module.Active = False
```

EDR Container Fields

Table 81-52 lists the FCT_Discount EDR container fields.

Table 81-52 FCT_Discount EDR Container Fields

Entry	Format	Access	Description
BDR_CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	Read	Used to calculate the duration.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Used to calculate the duration.
CREDIT_LIMIT_CHECK DETAIL.CREDIT_LIMIT_CHECK	Integer	Read	Determines whether the module applies discounts normally or as part of a credit limit check. If set to **1**, the module applies discounts as part of a credit limit check. It set to **0** (or any value other that **1**), the module operates normally.
ERROR_REJECT_TYPE DETAIL.ERROR_REJECT_TYPE	String	Read	Used by FCT_Reject to reject the DETAIL to a stream other than the standard reject stream.
EVENT_TYPE DETAIL.EVENT_TYPE	String	Read	BRM event type.
EVENT_BALANCE_GROUP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	Read	POID of the balance group charged.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
INTERN_PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Process status. If set to **2**, a recycle test is in progress, and this container is skipped.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	Date	Read	Usage class. This is used for matching the usage class in the discount detail.
REFRESH_BALANCE DETAIL.REFRESH_BALANCE	Integer	Read	Specifies whether the latest balance information should be retrieved from the database. When this field is set, this module calls the balance module to get the latest balance information from the database, whether or not a balance packet is present in the EDR.
USAGE_TYPE DETAIL.USAGE_TYPE	Date	Read	Usage type. Used for matching the usage type in the discount detail.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	UTC time offset, if the discount owner is not the A customer.
FU_DISCOUNT_OBJECTS DETAIL.ASS_CBD.FU_DISCOUNT_OBJECTS	String	Write	ID of the account's discounts that have first-usage start times which were used to discount the event.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Read	Record type. Used for matching the record type in the discount detail.
CHARGED_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read	Currency of the charge.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read	Amount of the charge. Used as the base value for discounting.
DISCOUNTMODEL_CODE DETAIL.ASS_CBD.CP.DISCOUNTMODEL_CODE	String	Read	Discount. Used for matching the discount in the discount detail.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Impact category Used for matching the impact category in the discount detail.
CP_DISCOUNT_PACKET_INDEX DETAIL.ASS_CBD.CP.INTERN_PACKET_INDEX	Integer	Read	Packet ID.
PRICEMODEL_CODE DETAIL.ASS_CBD.CP.PRICEMODEL_CODE	String	Read	Pricing code. Used for matching the pricing in the discount detail.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
QUANTITY_FROM DETAIL.ASS_CBD.CP.QUANTITY_FROM	Decimal	Read	Charge packet start quantity. Used to determine whether to split charge packets.
QUANTITY_TO DETAIL.ASS_CBD.CP.QUANTITY_TO	Decimal	Read	Charge packet end quantity. Used to determine whether to split charge packets.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read	Charge code. Used for filtering in the discount detail.
RATETAG_CODE DETAIL.ASS_CBD.CP.RATETAG_CODE	String	Read	Concatenation of charge definition. Used for filtering in the discount detail.
RESOURCE DETAIL.ASS_CBD.CP.RESOURCE	String	Read	Balance Element code. Used for filtering in the discount detail.
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Read	Numeric ID of the balance element. Used for filtering in the discount detail.
ROUNDED_QUANTITY_FROM DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_FROM	Decimal	Read	From quantity after rounding.
ROUNDED_QUANTITY_TO DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_TO	Decimal	Read	To quantity after rounding.
ROUNDED_QUANTITY_VALUE DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_VALUE	Decimal	Read	Rounded quantity. Used as the base value for discounting.
RUM DETAIL.ASS_CBD.CP.RUM	String	Read	RUM name. Used for matching the RUM in the discount detail.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Read	Service class. Used for matching the service class in the discount detail.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Read	Service code. Used for matching the service code in the discount detail.
TIMEMODEL_CODE DETAIL.ASS_CBD.CP.TIMEMODEL_CODE	String	Read	Time model code. Used for matching the time model in the discount detail.
TIMEZONE_CODE DETAIL.ASS_CBD.CP.TIMEZONE_CODE	String	Read	Time zone code. Used for matching the time zone in the discount detail.
USAGE_GL_ACCOUNT_CODE DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Read	G/L code. Used for matching the G/L code in the discount detail.
ASS_CBD_ZONEMODEL_CODE DETAIL.ASS_CBD.CP.ZONEMODEL_CODE	String	Read	Zone model code. Used for matching the zone model in the discount detail.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DETAIL.ASS_CBD.CP.SPLIT_CP	Sub-block	Write	Optional sub-block added when charge packets are split.
DETAIL.ASS_CBD.CP.SPLIT_CP.RESOURCE_ID	Integer	Write	Numeric ID of the balance element. Used for filtering in the discount detail. Copied from the original charge packet.
DETAIL.ASS_CBD.CP.SPLIT_CP.RUM	String	Write	RUM name. Copied from the original charge packet.
DETAIL.ASS_CBD.CP.SPLIT_CP.QUANTITY_FROM	Decimal	Write	Split charge packet start quantity. Calculated by the module.
DETAIL.ASS_CBD.CP.SPLIT_CP.QUANTITY_TO	Decimal	Write	Split charge packet end quantity. Calculated by the module.
DETAIL.ASS_CBD.CP.SPLIT_CP.CHARGED_AMOUNT_VALUE	Decimal	Write	Amount of the charge for this split charge packet. Calculated by the module.
DETAIL.ASS_CBD.CP.SPLIT_CP.INTERN_PACKET_INDEX	Integer	Write	The index of the split charge packet.
DETAIL.ASS_CBD.CP.SPLIT_CP.INTERN_SRC_PACKET_INDEX	Integer	Write	The packet index of the charge packet from which this split charge packet was generated.
DP_DISCOUNT_BALANCE_GROUP_ID DETAIL.ASS_CBD.DP.BALANCE_GROUP_ID	String	Write	POID of the balance group impacted by this discount packet.
DP_CREATED DETAIL.ASS_CBD.DP.CREATED	String	Write	Creation date of the element.
DP_DISCOUNTBALIMPACTID DETAIL.ASS_CBD.DP.DISCOUNTBALIMPACTID	Integer	Write	Discount balance impact ID.
DP_DISCOUNTMODEL DETAIL.ASS_CBD.DP.DISCOUNTMODEL	String	Write	Discount used to create this discount packet.
DP_DISCOUNTRULE DETAIL.ASS_CBD.DP.DISCOUNTRULE	String	Write	Discount rule used to create this discount packet
DP_DISCOUNTSTEPID DETAIL.ASS_CBD.DP.DISCOUNTSTEPID	Integer	Write	Discount step used to create this discount packet
DP_DISCOUNT_GLID DETAIL.ASS_CBD.DP.GLID	String	Write	G/L ID as specified in the discount balance impact, otherwise copied from the charge packet.
DP_GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Write	Granted discount/ sponsorship amount. Can be currency or noncurrency.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DP_GRANTED_QUANTITY DETAIL.ASS_CBD.DP.GRANTED_QUANTITY	Decimal	Write	Discount base value used to compute granted amount.
DP_DISCOUNT_IMPACT_CATEGORY DETAIL.ASS_CBD.DP.IMPACT_CATEGORY	String	Write	Impact category specified for this discount packet, otherwise copied from the charge packet.
DP_INTERN_DISC_MATCH_FACTOR DETAIL.ASS_CBD.DP.INTERN_DISC_MATCH_FACTOR	Decimal	Write	Discount match factor
DP_DISCOUNT_PACKET_INDEX DETAIL.ASS_CBD.DP.INTERN_PACKET_INDEX	Integer	Write	Packet ID.
DP_DISCOUNT_SRC_PACKET_INDEX DETAIL.ASS_CBD.DP.INTERN_SRC_PACKET_INDEX	Integer	Write	Source packet ID.
DP_INTERN_TOTAL_MATCH_FACTOR DETAIL.ASS_CBD.DP.INTERN_TOTAL_MATCH_FACTOR	Decimal	Write	Total discounted match factor.
DP_NODE_LOCATION DETAIL.ASS_CBD.DP.NODE_LOCATION	String	Write	Node location.
DP_OBJECT_ACCOUNT DETAIL.ASS_CBD.DP.OBJECT_ACCOUNT	Integer	Write	POID of discount owner.
DP_OBJECT_ID DETAIL.ASS_CBD.DP.OBJECT_ID	String	Write	Discount/sponsor object ID.
DP_OBJECT_OWNER_ID DETAIL.ASS_CBD.DP.OBJECT_OWNER_ID	Integer	Write	POID of the account or service that owns the discount.
DP_OBJECT_OWNER_TYPE DETAIL.ASS_CBD.DP.OBJECT_OWNER_TYPE	String	Write	POID type of discount owner, **/account** or **/service**.
DP_OBJECT_TYPE DETAIL.ASS_CBD.DP.OBJECT_TYPE	String	Write	Discount/sponsor object that generated this discount.
DP_DISCOUNT_PRICEMODEL_CODE DETAIL.ASS_CBD.DP.PRICEMODEL_CODE	String	Write	Pricing used to generate this discount.
DP_DISCOUNT_QUANTITY DETAIL.ASS_CBD.DP.QUANTITY	Decimal	Write	Discounted noncurrency amount.
DP_QUANTITY_FROM DETAIL.ASS_CBD.DP.QUANTITY_FROM	Decimal	Write	Discount packet start quantity. Aligns with the QUANTITY_FROM value in a charge packet or a split charge packet.
DP_QUANTITY_TO DETAIL.ASS_CBD.DP.QUANTITY_TO	Decimal	Write	Discount packet end quantity. Aligns with the QUANTITY_TO value in a charge packet or a split charge packet.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DP_DISCOUNT_RATEPLAN DETAIL.ASS_CBD.DP.RATEPLAN	String	Write	Charge used to generate this discount.
DP_DISCOUNT_RATETAG DETAIL.ASS_CBD.DP.RATETAG	String	Write	Concatenation of the definition of the charge used to generate this discount.
DP_DISCOUNT_RESOURCE DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Write	Balance Element ID of the balance element impacted by this discount.
DP_DISCOUNT_RUM DETAIL.ASS_CBD.DP.RUM	String	Write	The RUM entered for filtering in the discount filter.
DP_DISCOUNT_SERVICE_CLASS DETAIL.ASS_CBD.DP.SERVICE_CLASS	String	Write	Service class entered for filtering in the discount filter.
DP_DISCOUNT_SERVICE_CODE DETAIL.ASS_CBD.DP.SERVICE_CODE	String	Write	Service code entered for filtering in the discount filter.
DP_DISCOUNT_SUB_BALANCE.GRANTOR DETAIL.ASS_CBD.DP.SUB_BALANCE.GRANTOR	String	Read	ID of the charge offer or discount that granted this balance element.
DP_DISCOUNT_SUB_BALANCE.VALID_FROM_DETAILS DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM_DETAILS	Integer	Read	The sub-balance start time mode (such as first-usage or relative) and relative offset details. This field is used in conjunction with SUB_BALANCE_VALID_FROM to determine the validity period start time.
DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO_DETAILS	Integer	Read	The sub-balance end time mode (such as relative) and relative offset details. This field is used in conjunction with SUB_BALANCE_VALID_TO to determine the validity period end time.
DP_DISCOUNT_TAX_CODE DETAIL.ASS_CBD.DP.TAX_CODE	String	Write	Tax code as specified in the discount balance impact. If not specified in the balance impact, copied from the charge packet for this discount packet.
DP_DISCOUNT_TIMEMODEL_CODE DETAIL.ASS_CBD.DP.TIMEMODEL_CODE	String	Write	Time model entered for filtering in the discount filter.
DP_DISCOUNT_TIMEZONE_CODE DETAIL.ASS_CBD.DP.TIMEZONE_CODE	String	Write	Time zone entered for filtering in the discount filter.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
DP_DISCOUNT_VALID_FROM DETAIL.ASS_CBD.DP.VALID_FROM	Date	Write	Valid-from date for the grant in the discount packet.
DP_DISCOUNT_VALID_TO DETAIL.ASS_CBD.DP.VALID_TO	Date	Write	Valid-to date for the grant in the discount packet.
DP_DISCOUNT_VALID_FROM_DETAIL DETAIL.ASS_CBD.DP.VALID_FROM_DETAIL	Integer	Read	The valid-from mode (such as first usage or relative) and relative offset details. This field is used in conjunction with PIN_FLD_VALID_FROM to determine the validity period start time.
DP_DISCOUNT_VALID_TO_DETAIL DETAIL.ASS_CBD.DP.VALID_TO_DETAIL	Integer	Read	The valid-to mode (such as relative) and relative offset details. This field is used in conjunction with PIN_FLD_VALID_TO to determine the validity period end time.
DP_DISCOUNT_ZONEMODEL_CODE DETAIL.ASS_CBD.DP.ZONEMODEL_CODE	String	Write	Zone model code enter for filtering in the discount filter.
SUB_BAL_AMOUNT DETAIL.ASS_CBD.DP.SUB_BALANCE.AMOUNT	Decimal	Write	Amount of a sub-balance impacted by the discount packet.
SUB_BAL_CONTRIBUTOR DETAIL.ASS_CBD.DP.SUB_BALANCE.CONTRIBUTOR	String	Write	Contributor to a sub-balance impacted by the discount packet.
SUB_BAL_REC_ID DETAIL.ASS_CBD.DP.SUB_BALANCE.REC_ID	Integer	Write	ID of a sub-balance impacted by the discount packet.
SUB_BAL_VALID_FROM DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_FROM	Date	Write	Beginning validity date for a sub-balance impacted by the discount packet.
SUB_BAL_VALID_TO DETAIL.ASS_CBD.DP.SUB_BALANCE.VALID_TO	Date	Write	End validity date for a sub-balance impacted by the discount packet.
ACCOUNT_PARENT_ID DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Customer account POID.
ACTG_LAST_DATE DETAIL.CUST_A.ACTG_LAST_DATE	Date	Read	The date that the current monthly cycle began.
ACTG_NEXT_DATE DETAIL.CUST_A.ACTG_NEXT_DATE	Date	Read	Date that the current monthly cycle ends.
ACTG_USED_DATE DETAIL.CUST_A.ACTG_USED_DATE	Date	Read	Date used for this EDR.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
FIRST_USGAE_INDICATOR DETAIL.CUST_A.DL.PD.FIRST_USAGE_INDICATOR	Integer	Read	Specifies whether the discount's validity period is configured to start when first used.
DETAIL.CUST_A.DL.PD.USAGE_END		Write	
INTERN_FOUND_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Purchased charge offer index of the charge offer or service.
SERVICE_ID DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	POID of the service instance.
BG_BG_ID DETAIL.CUST_A.BG.BALANCE_GROUP_ID	String	Read	POID of the balance group for the account.
BG_BELEM_CURR_BAL DETAIL.CUST_A.BG.BAL_ELEM.CURR_BAL	Decimal	Read	Current balance of the balance group element.
BG_BELEM_RESOURCE_ID DETAIL.CUST_A.BG.BAL_ELEM.RESOURCE_ID	Integer	Read	Balance Element ID of the balance group element.
DISCOUNT_BALANCE_GROUP_ID DETAIL.CUST_A.DL.BALANCE_GROUP_ID	String	Read	Balance group used to evaluate this discount instance.
DISCOUNT_OWNER_ACCT_ID DETAIL.CUST_A.DL.DISCOUNT_OWNER_ACCT_ID	String	Read	POID of the account that owns the discount, either directly or indirectly through a service.
DISOUNT_OWNER_ID DETAIL.CUST_A.DL.DISCOUNT_OWNER_ID	String	Read	POID of the account or service that owns the discount. Identical to DISCOUNT_OWNER_ACCT_ID if the discount is owned directly by an account.
DISOUNT_OWNER_TYPE DETAIL.CUST_A.DL.DISCOUNT_OWNER_TYPE	String	Read	Discount owner type, either **/account** or **/service**.
PD_DISCOUNTID DETAIL.CUST_A.DL.PD.DISCOUNT_ID	String	Read	POID of the discount object.
PD_DISCOUNTMODEL DETAIL.CUST_A.DL.PD.DISCOUNT_MODEL	String	Read	Code of a discount referenced in the discount object.
PD_FLAGS DETAIL.CUST_A.DL.PD.FLAGS	Integer	Read	Proration setting.
PD_MODE DETAIL.CUST_A.DL.PD.MODE	Integer	Read	Discount mode, either cascading or parallel.
PD_NODE_LOCATION DETAIL.CUST_A.DL.PD.NODE_LOCATION	String	Read	Unique ID of the discount object.

Table 81-52 (Cont.) FCT_Discount EDR Container Fields

Entry	Format	Access	Description
PD_QUANTITY DETAIL.CUST_A.DL.PD.QUANTITY	Integer	Read	Number of purchased discounts. This is multiplied by balance impact of this discount instance.
PD_SCALE DETAIL.CUST_A.DL.PD.SCALE	Decimal	Read	Proration scale.
PD_VALID_FLAG DETAIL.CUST_A.DL.PD.VALID_FLAG	Integer	Read	Indicates whether the discount is valid.
SPONSOR_BALANCE_GROUP_ID DETAIL.CUST_A.SL.BALANCE_GROUP_ID	String	Read	POID of the charge share group balance group.
SPONSOR_OWNER_ACCT_ID DETAIL.CUST_A.SL.SPONSOR_OWNER_ACCT_ID	String	Read	POID of the account that owns the chargeshare object, either directly or indirectly through a service.
SPONSOR_OWNER_ID DETAIL.CUST_A.SL.SPONSOR_OWNER_ID	String	Read	POID of the account or service that owns the chargeshare. Identical to SPONSOR_OWNER_ACCT_ID if the chargeshare is owned directly by an account.
SPONSOR_OWNER_TYPE DETAIL.CUST_A.SL.SPONSOR_OWNER_TYPE	String	Read	Chargeshare owner type, either **/account** or **/service**.
SD_DISCOUNTMODEL DETAIL.CUST_A.SL.SD.DISCOUNT_MODEL	String	Read	Discount used for this chargeshare.
SD_SPONSORID DETAIL.CUST_A.SL.SD.SPONSORSHIP_ID	String	Read	The POID of the chargeshare (**/sponsorship**) object.
SD_VALID_FLAG DETAIL.CUST_A.SL.SD.VALID_FLAG	Integer	Read	Indicates whether the chargeshare is valid.
DETAIL.CUST_A.SUBCYCLE_PL.DL.PD.USAGE_END	N/A	Read	N/A
DETAIL.CUST_A.SUBCYCLE_PL.DL.PD.FIRST_USAGE_INDICATOR	N/A	Read	N/A
TRANSACTION_ID INTERNAL.TRANSACTION_ID	Decimal	Read	The transaction ID. Needed for queuing.

FCT_DiscountAnalysis

The FCT_DiscountAnalysis module determines which discounts apply to a given event.

See "Configuring Discounting Modules and Components".

Dependencies

The FCT_DiscountAnalysis module requires a connection to the following data modules:

- DAT_Discount
- DAT_ModelSelector

For pipeline rating, this module must run after the FCT_Account module and before the FCT_Discount module.

For real-time rating, this module must run before the FCT_Discount module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-53 lists the FCT_DiscountAnalysis registry entries.

Table 81-53 FCT_DiscountAnalysis Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DiscountModelDataModule	Specifies the connection to the DAT_Discount module.	Yes
Filter_SetModule	Specifies the connection to the FCT_Filter_Set module. Use this entry if the FCT_Filter_Set module is configured. See: • About Using Filter Sets to Apply System Products and Discounts • FCT_Filter_Set	No
ModelSelectorDataModule	Specifies the connection to the DAT_ModelSelector module.	Yes

Sample Registry

```
Discount
{
    ModuleName = FCT_DiscountAnalysis
    Module
    {
       Active = True
       DiscountModelDataModule = ifw.DataPool.DiscountModelDataModule
       Filter_SetModule = ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Filter_Set
       ModelSelectorDataModule = ifw.DataPool.ModelSelectorDataModule
    }
}
```

Semaphore File Entries

Table 81-54 lists the FCT_DiscountAnalysis Semaphore file entry.

Table 81-54 FCT_DiscountAnalysis Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.DiscountAnalysisModule.
Module.Active = False
```

EDR Container Fields

Table 81-55 lists the FCT_DiscountAnalysis EDR container fields.

Table 81-55 FCT_DiscountAnalysis EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	R	n/a
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	R	Charge packet; used to check for any discount ERAs set.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	R	Event start time.
BDR_CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	R	Event end time.
EVENT_TYPE DETAIL.EVENT_TYPE	String	R	The event type used to locate the event discount.
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	R	The UTC offset used to adjust the start and end time.
INTERN_BALANCE_GROUP_ID DETAIL.INTERN_BALANCE_GROUP_ID	String	RW	Account level balance group of the event owner account.
INTERN_DISCOUNT_OWNER_ACCT_ID DETAIL.INTERN_DISCOUNT_OWNER_ACCT_ID	String	RW	Event owner account ID.
DISCOUNT_LIST DETAIL.CUST_A.DL	Block	R	Purchased discount offers that belong to an account or service, or are shared by an account or service.
BALANCE_GROUP_ID DETAIL.CUST_A.DL.BALANCE_GROUP_ID	String	RW	ID of the balance group whose balance elements are used for the discounts in the discount list.

Table 81-55 (Cont.) FCT_DiscountAnalysis EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DISCOUNT_OWNER_ACCT_ID DETAIL.CUST_A.DL.DISCOUNT_OWNER_ACCT_ID	String	RW	ID of the account that owns the set of purchased discount offers in the discount list.
PURCHASED_DISCOUNTS DETAIL.CUST_A.DL.PD	String	CW	Information about the discount.
DISCOUNT_ID DETAIL.CUST_A.DL.PD.DISCOUNT_ID	String	RW	Discount Ooffer object ID.
STATUS DETAIL.CUST_A.DL.PD.STATUS	String	RW	Discount offer state (active, inactive, or cancelled).
PURCHASE_START DETAIL.CUST_A.DL.PD.PURCHASE_START	Date	RW	Discount offer purchase start time.
PURCHASE_END DETAIL.CUST_A.DL.PD.PURCHASE_END	Date	RW	Discount offer purchase end time.
USAGE_START DETAIL.CUST_A.DL.PD.USAGE_START	Date	RW	Discount usage start time.
USAGE_END DETAIL.CUST_A.DL.PD.USAGE_END	Date	RW	Discount usage end time.
PRIORITY DETAIL.CUST_A.DL.PD.PRIORITY	Integer	RW	Discount offer priority.
MODE DETAIL.CUST_A.DL.PD.MODE	Integer	RW	Discount mode (parallel or cascading).
VALID_FLAG DETAIL.CUST_A.DL.PD.VALID_FLAG	Integer	RW	A value indicating discount validity.
TYPE DETAIL.CUST_A.DL.PD.TYPE	Integer	RW	Discount type (system, subscription, or item).
QUANTITY DETAIL.CUST_A.DL.PD.QUANTITY	Decimal	RW	Purchase quantity.
SCALE DETAIL.CUST_A.DL.PD.SCALE	Decimal	RW	Proration scale.
OFFERING_POID DETAIL.CUST_A.DL.PD.OFFERING_POID	String	RW	A value that identifies the purchased discount associated with the account.
DISCOUNT_MODEL DETAIL.CUST_A.DL.PD.DISCOUNT_MODEL	String	RW	A discount.
SPONSOR_LIST DETAIL.CUST_A.SL	Block	R	The list of sponsors that split the charges with the event user.
SPONSOR_BALANCE_GROUP_ID DETAIL.CUST_A.SL.BALANCE_GROUP_ID	String	R	The balance group whose balance elements are used for the sponsorship list.

Table 81-55 (Cont.) FCT_DiscountAnalysis EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
SPONSORSHIP_DETAILS DETAIL.CUST_A.SL.SD	Block	R	Sponsorships that belong to an account or service or are shared by the account or service.
SPONSORSHIP_ID DETAIL.CUST_A.SL.SD.SPONSORSHIP_ID	String	R	Sponsorship object ID.
SPONSOR_VALID_FLAG DETAIL.CUST_A.SL.SD.VALID_FLAG	Integer	RW	A value that indicates sponsorship validity.
SPONSOR_DISCOUNT_MODEL DETAIL.CUST_A.SL.SD.DISCOUNT_MODEL	String	RW	Sponsorship model.

FCT_DroppedCall

The FCT_DroppedCall module identifies phone calls that were terminated due to technical reasons and then resumed again through a customer's subsequent phone call.

Dependencies

Run the FCT_DroppedCall module *after* the FCT_Account module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-56 lists the FCT_DroppedCall registry entries.

Table 81-56 FCT_DroppedCall Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
FilePath	Specifies the path to the dropped calls data file. The default is the data directory (**./data**).	No
FileName	Specifies the name of the dropped calls data file, which stores back-up information about dropped calls. You use this file to restore information in case of system error or system restart.	Yes
TempPrefix	Specifies the prefix for the temporary dropped calls data files. The default is **tmp_**.	No
CheckField	Section that specifies the EDR field and values used to identify a dropped call.	Yes
CheckField.Name	Specifies the EDR field that is used to identify a dropped call. **Note:** Only one EDR field can be used to identify a dropped call.	Yes

Table 81-56 (Cont.) FCT_DroppedCall Registry Entries

Entry	Description	Mandatory
CheckField.Value	Specifies the values for identifying a dropped call. **Note:** If more than one value qualifies an EDR as a dropped call, enter multiple values separated by a comma (,) with no spaces; for example: 5,6,7. BRM interprets the comma as a Boolean OR value.	Yes
WrittenFields	Section that specifies the dropped call EDR fields that are written into memory and are used to detect continuation calls. You use dummy key values such as 1 and 2 to list the EDR fields, as shown below: **1 =** *EDR_field* **2 =** *EDR_field* **3 =** *EDR_field* **Note:** BRM automatically writes the DETAIL.A_NUMBER, DETAIL.B_NUMBER, DETAIL.CHARGING_END_TIMESTAMP, and DETAIL.CUST_A.BILL_NEXT_DATE EDR fields to memory, so you should not list these fields.	No
AddedFields	Section that specifies the dropped call EDR fields that are written into memory and then added to the continuation call EDR. Important: When you map a dropped call EDR field to a continuation call EDR field, both fields must have the same data type. You can find a field's data type by reading the container description file (**container.dsc**). By default, the module does not enrich the continuation call EDR.	No
AddedFields.Field*x*	Section that maps one dropped call EDR field to one continuation call EDR field. You create a **Field***x* section for each pair of EDR fields that you want to map. For example, if you want to map three EDR pairs, create a **Field1** section, a **Field2** section, and a **Field3** section.	No
AddedFields.Field*x***.ContinuationCallField**	Specifies the continuation call EDR field. The module adds the value of the dropped call EDR field specified in **DroppedCallField** to the continuation call EDR field that you specify.	No
AddedFields.Field*x***.DroppedCallField**	Specifies the dropped call EDR field to write into memory. This field's value is added to the continuation call EDR field specified in **ContinuationCallField**.	No

Sample Registry

```
DroppedCall
{
  ModuleName = FCT_DroppedCall
  Module
  {
    Active = TRUE
    FilePath = ./data
    FileName = dropped_call
    TempPrefix = tmp_
    CheckField
    {
      Name = DETAIL.CALL_COMPLETION_INDICATOR
      Value = 5
    }
    WrittenFields
    {
      1 = DETAIL.RECORD_TYPE
    }
    AddedFields
```

```
    {
      Field1
      {
        ContinuationCallField = DETAIL.DROPPED_CALL_QUANTITY
        DroppedCallField = DETAIL.DURATION
      }
    }
  }
}
```

With this sample registry configuration, the FCT_DroppedCall module identifies:

- *Dropped calls* by finding all EDRs with a DETAIL.CALL_COMPLETION_INDICATOR EDR field set to **5**.
- *Continuation calls* by using the DETAIL.RECORD_TYPE EDR field.

> **Note:**
>
> By default, the module also automatically writes the DETAIL.A_NUMBER, DETAIL.B_NUMBER, DETAIL.CUST_A.BILL_NEXT_DATE, and DETAIL.CHARGING_END_TIMESTAMP EDR fields to memory.

When the module detects a continuation call, it adds the value of the dropped call's DETAIL.DURATION EDR field to the continuation call's DETAIL.DROPPED_CALL_QUANTITY EDR field.

Semaphore File Entries

Table 81-57 lists the FCT_DroppedCall Semaphore file entries.

Table 81-57 FCT_DroppedCall Semaphore File Entries

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive
RemoveLimit	Specifies to remove all calls from the memory map and data file that are older than the specified number of days. For example, if you specify **7**, BRM removes from the memory map all entries older than 7 days. The time is calculated from the current system time. **Note:** Set the time to a large enough value to allow for late-arriving and recycled EDRs.

Sample Semaphore File Entries

```
ifw.Pipelines.ALL_RATE.Functions.FunctionPool.DroppedCall.Active = True
ifw.Pipelines.ALL_RATE.Functions.FunctionPool.DroppedCall.RemoveLimit = 7
```

EDR Container Fields

The FCT_DroppedCall module accesses the EDR container fields shown in Table 81-58. You can configure the module to access additional EDR container fields by using the module's **CheckField**, **WrittenFields**, and **AddedFields** registry entries.

Table 81-58 FCT_DroppedCall EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the customer A number.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the customer B number.
CUST_A_BILL_NEXT_DATE DETAIL.CUST_A.BILL_NEXT_DATE	Date	Read	Contains the timestamp for the customer's next billing cycle.
CHARGING_END_TIMESTAMP DETAIL.CHARGING_END_TIMESTAMP	Date	Read	Contains the dropped call's ending timestamp.
DURATION DETAIL.DURATION	Decimal	Read	Contains the duration of the current call.
CUST_A_PROFILE DETAIL.CUST_A.ERA.PROFILE	String	Read	Contains the customer's account-related ERA data.
CUST_A_KEY DETAIL.CUST_A.ERA.PA.KEY	String	Read	Contains the key for the account-related ERA data.
CUST_A_VALUE DETAIL.CUST_A.ERA.PA.VALUE	String	Read	Contains the value for the account-related ERA data.
CALL_COMPLETION_INDICATOR DETAIL.CALL_COMPLETION_INDICATOR	String	Read	Contains the reason the current call session was terminated.
DROPPED_CALL_STATUS DETAIL.DROPPED_CALL_STATUS	Integer	Write	Flags the status of the call: **0** = Normal call **1** = Dropped call **2** = Continuation call **3** = Both a dropped call and a continuation call **4** = A call that was processed by FCT_DroppedCall but didn't qualify as a dropped call or a continuation call.
DROPPED_CALL_QUANTITY DETAIL.DROPPED_CALL_QUANTITY	Decimal	Write	When the EDR is flagged as a continuation call, this field contains the duration of the associated dropped call.

FCT_DuplicateCheck

The FCT_DuplicateCheck module checks for duplicate EDRs. See "Handling Duplicate EDRs".

> **Note:**
>
> Before using the FCT_DuplicateCheck module, load the duplicate check stored procedures in the Pipeline Manager database.

Dependencies

To enable your system to check for duplicate EDRs without using excessive disk space, connect the FCT_DuplicateCheck module to the Pipeline Manager database.

The FCT_DuplicateCheck module is typically the second module in a pipeline, directly following the FCT_PreSuspense module. This ensures that no further processing is done on duplicate EDRs.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-59 lists the FCT_DuplicateCheck registry entries.

Table 81-59 FCT_DuplicateCheck Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
BufferLimit	Specifies the oldest date that previously processed EDRs can be stored in memory. The format is *YYYYMMDD*. Note: The **BufferLimit** date must be equal to or later than the **StoreLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later. You can use this entry in a semaphore file. See "Setting Date Parameters for Storing Processed EDRs".	Yes
BulkInsertArraySize	Specifies the maximum number of rows for bulk insert when the data in memory flushes to the database. Default = **10000**.	No
DataConnection	Specifies a connection to the Pipeline Manager database. **Important:** To avoid using excessive disk space when checking for duplicate EDRs, enable this entry.Default = **False**. See "About Storing EDRs in a Database Instead of Files".	No
DuplicateIndicatorField	Specifies the EDR field to set if an EDR is a duplicate. The field specified must be an integer field. You can use any integer field in the EDR. This entry is used by the FCT_CiberOcc module to determine whether to create an OCC record. OCC records are not created for duplicate EDRs.	No
Fields	Specifies the EDR fields that are used for checking. **Important:** Do not use the DETAIL_CHARGING_START_TIMESTAMP field for duplicate checking. See "Specifying the Fields to Use for Duplicate Check".	Yes

Table 81-59 (Cont.) FCT_DuplicateCheck Registry Entries

Entry	Description	Mandatory
FileName	Specifies the base file name of the data files (the transaction ID and suffix are appended). Default = **.** See "Managing FCT_DuplicateCheck Data Files".	Yes
IndexSpaceName	Index space name where the run-time duplicate check index is created (database mode only). For example: `IndexSpaceName = INTEGRATE_TS_1_IDX`	Yes, if using a database connection.
Path	Specifies the directory for the data files that store EDRs. See "Managing FCT_DuplicateCheck Data Files".	No
SearchKey	Identifies duplicate EDRs. See "Specifying a Search Key for Duplicate Check". **Important:** If you use the **SearchKey** registry entry, don't list the **SearchKey** value as a field in the **Fields** list.	No
StoreLimit	Specifies the oldest date that previously processed EDRs can be stored. If an EDR is dated earlier than the **StoreLimit** date, the EDR is not processed by the FCT_DuplicateCheck module. The format is *YYYYMMDD*. **Note:** The **StoreLimit** date must be equal to or earlier than the **BufferLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later.You can use this entry in a semaphore file. See "Setting Date Parameters for Storing Processed EDRs".	Yes
StreamName	Specifies the output stream for duplicate EDRs. See "Configuring Output for Rejected or Duplicate EDRs".	Yes
TableSpaceName	Table space name where the run-time duplicate check table is created (database mode only). When a **StoreLimit** or **BufferLimit** semaphore is sent, FCT_DuplicateCheck needs to know where to store data. This entry, and the **IndexSpaceName** entry specify the location in the database. For example: `TableSpaceName = INTEGRATE_TS_1_DAT`	Yes, if using a database connection.
TableSuffix	Allows you to create multiple IFW_DUPLICATECHECK tables when you run multiple pipelines.	No

Sample Registry

> **Note:**
>
> When entering data in the **Fields** entry, use dummy key values such as **1**, **2**, and **3**, as shown in this example.

```
DuplicateCheck
{
   ModuleName  =  FCT_DuplicateCheck
   Module
   }
      Active  =  True
      DataConnection  =  ifw.DataPool.DupLogin1
      Path  =  ./data/dup
      Filename  =  call.duplicate
      StreamName  =  DuplicateOutput
      BufferLimit  =  20040105
      StoreLimit  =  20040101
      SearchKey  =  DETAIL.A_NUMBER
      Fields
      {
        1  =  DETAIL.BASIC_SERVICE
        2  =  DETAIL.B_NUMBER
      }
   }
}
```

With this sample registry configuration, the following occurs:

- EDRs dated January 5, 2004 (**BufferLimit**) or later are stored in the duplicate list in memory.
- EDRs dated January 1, 2004 (**StoreLimit**) through January 4, 2004 are stored in the IFW_DUPLICATECHECK database table.
- EDRs dated December 31, 2003 or earlier are ignored.

The following day, the module receives the following update registry:

```
DuplicateCheck
{
   ModuleName  =  FCT_DuplicateCheck
   Module
   {
      BufferLimit  =  20040106
      StoreLimit  =  20040102
   }
}
```

With this updated registry sample configuration, the following occurs:

- EDRs dated January 6, 2004 (new **BufferLimit**) or later are stored in the duplicate check list in memory.
- EDRs dated January 5, 2004 are moved to the IFW_DUPLICATECHECK database table.
- EDRs dated January 2, 2004 (new **StoreLimit**) through January 4, 2004 continue to be stored in the database table.

> **Note:**
>
> EDR data for duplicate checks is stored in the IFW_DUPLICATECHECK database table. This table can be hosted by any database. Normally, at the end of the day, all the EDR data in memory is flushed to the database.

Semaphore File Entries

Table 81-60 lists the FCT_DuplicateCheck Semaphore file entries.

Table 81-60 FCT_DuplicateCheck Semaphore File Entries

Entry	Description
Active	Specifies whether the module is active or inactive. **True** = Active **False** = Inactive
BufferLimit	Specifies the oldest date that previously processed EDRs can be stored in memory. The format is *YYYYMMDD*. **Note:** The **BufferLimit** date must be equal to or later than the **StoreLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later. See "Setting Date Parameters for Storing Processed EDRs".
StoreLimit	Specifies the oldest date that previously processed EDRs can be stored. If an EDR is dated earlier than the **StoreLimit** date, the EDR is not processed by the FCT_DuplicateCheck module. The format is *YYYYMMDD*. **Note:** The **BufferLimit** date must be equal to or later than the **StoreLimit** date. For example, if the **StoreLimit** date is June 1, the **BufferLimit** must be June 1 or later. See "Setting Date Parameters for Storing Processed EDRs".

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.DuplicateCheck.
Module.StoreLimit = 20020101

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.DuplicateCheck.
Module.BufferLimit = 20020125
```

Sample Output Configuration

You configure the output stream for the FCT_DuplicateCheck module in the Output section of the registry, for example:

```
# Output stream for duplicate events
DuplicateOutput
{
 ModuleName = OUT_Reject
 Module
 {
  OutputStream
  {
   ModuleName = EXT_OutFileManager
   Module
   {
    OutputPath = ./samples/wireless/data/rej
    OutputPrefix = test
    OutputSuffix = .dup
    TempPrefix   = tmp

    TempDataPath    = ./samples/wireless/data/rej
    TempDataPrefix  = dup.tmp.
```

```
    TempDataSuffix  = .data

    Replace = TRUE
    DeleteEmptyFile = TRUE
   }
  }
 }
} # end of DuplicateOutput
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

EDR Container Fields

You specify the EDR container fields in the FCT_DuplicateCheck module **Fields** startup registry entry.

Database Tables

The FCT_DuplicateCheck module uses the IFW_DUPLICATECHECK database table.

If you use the database instead of a file to define the data that the FCT_DuplicateCheck module uses for comparing EDRs, you need to create this table. The FCT_DuplicateCheck module uses the data in the IFW_DUPLICATECHECK table to check for duplicate EDRs. See "Handling Duplicate EDRs".

The IFW_DUPLICATECHECK table should have a unique index. A duplicate EDR is detected by the database reporting a violation of the uniqueness when attempting to INSERT.

> **Note:**
>
> Oracle recommends that you have multiple partitions to increase the INSERT performance.

FCT_EnhancedSplitting

The FCT_EnhancedSplitting module specifies different output streams for EDRs based on rules. For example:

- You can split EDRs for different service types to different output streams.
- You can split EDRs from roaming outcollects and incollects into different streams.

See "Using Rules to Send EDRs to Different Output Streams".

Dependencies

Requires a connection to the Pipeline Manager database.

Because you can split EDRs based on service codes, this module should run after the FCT_ServiceCodeMap and FCT_UsageClassMap modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-61 lists the FCT_EnhancedSplitting registry entries.

Table 81-61 FCT_EnhancedSplitting Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
DefaultOutput	Specifies the default output stream if no splitting rule matches the current EDR. If no default output stream is specified, the EDR receives the error message ERR_NO_SPLITTING_PERFORMED. Default = **.** See "Configuring EDR Output Processing".	No
SystemBrands	Maps the system brand table to the output stream. Each entry in this section has the format SYSBRAND=OUTPUT-STREAM. See "Using Rules to Send EDRs to Different Output Streams".	Yes

Sample Registry

```
Splitting
{
   ModuleName = FCT_EnhancedSplitting
   Module
      {
      Active = True
      DataConnection = Login
      DefaultOutput = EdrOutput0
      SystemBrands
         {
         1 = EdrOutput1
         2 = EdrOutput2
      }
   }
}
```

Semaphore File Entries

Table 81-62 lists the FCT_EnhancedSplitting Semaphore file entries.

Table 81-62 FCT_EnhancedSplitting Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database.

Sample Semaphore File Entry

```
ifw.Pipelines.PRE_PRODCESS.Functions.PreProcessing.FunctionPool.
EnhancedSplitting.Module.Active = True
```

EDR Container Fields

Table 81-63 lists the FCT_EnhancedSplitting EDR container fields.

Table 81-63 FCT_EnhancedSplitting EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network code. This could either be the PLMN ID or any logical operator code.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network code.
ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GSM MSC or Switch ID handling the origin of the call.
ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION DETAIL.ASS_GPRS_EXT.ORIGINATING_SWITCH_IDENTIFICATION	String	Read	Contains the GPRS MSC or Switch ID handling the origin of the call.
ASS_GSMW_EXT.TRUNK_INPUT DETAIL.ASS_GSMW_EXT.TRUNK_INPUT	String	Read	Contains the trunk identification (in-route address in network switches).
ASS_GSMW_EXT_TRUNK_OUTPUT DETAIL.ASS_GSMW_EXT.TRUNK_OUTPUT	String	Read	Contains the trunk identification (out-route address in network switches).

Table 81-63 (Cont.) FCT_EnhancedSplitting EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the A number.
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number.
INTERN_C_NUMBER_ZONE DETAIL.INTERN_C_NUMBER_ZONE	String	Read	Contains the number where the call was first terminated if it was forwarded or routed.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the time-stamp used for start of charging.

Database Tables

The FCT_EnhancedSplitting module uses the following database tables:

- IFW_SPLITTING_TYPE. The FCT_EnhancedSplitting module uses the data in the IFW_SPLITTING_TYPE table to determine how to route EDRs to different output streams based on rules. See "Using Rules to Send EDRs to Different Output Streams".
- IFW_SYSTEM_BRAND. The IFW_SYSTEM_BRAND table stores the system brand codes. See "Using Rules to Send EDRs to Different Output Streams".

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_EventOrder

When an event is rated, the FCT_EventOrder module uses the criteria and the event timestamps to determine if the event needs to be rerated.

Dependencies

This module must run *after* the FCT_MainRating and FCT_Discount modules and *before* the FCT_Reject module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-64 lists the FCT_EventOrder registry entries.

Table 81-64 FCT_EventOrder Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive: **True** = Active **False** = Inactive	Yes
AccountDataModule	Specifies the connection to the DAT_AccountBatch module.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module.	Yes
RerateDelayTolerance	Specifies a time in minutes that controls how much out-of-order EDR data FCT_EventOrder writes to a rerate-request file before creating a new file.	No
NumberOfAccountLimit	Specifies the number of accounts FCT_EventOrder assigns to each rerate job. This entry affects batch rerating throughput.	No
OutputDirectory	Specifies the output directory for out-of-order events. **Important:** You must create this directory. It is not created by BRM installation scripts.	Yes
OutputPrefix	Specifies the prefix of the rerate-request file name. The default is **ood**.	Yes
SkipPrevBillingCycle	Specifies whether to skip events that belong to previous billing cycles. **True** = Skip the events **False** = Do not skip the events (Default)	No

Sample Registry

```
EventOrder
{
  ModuleName                = FCT_EventOrder
  Module
  {
    Active                  = True
    AccountDataModule       = ifw.DataPool.CustomerData
    PortalConfigDataModule  = ifw.DataPool.PortalConfigDataModule

#delay tolerance in minutes for creating rerate jobs
    RerateDelayTolerance    =  180

#maximum number of accounts in the rerate-request file
#this should match the value of the pin_rerate "per_job"
#configuration entry
    NumberOfAccountLimit    = 1000

#output directory and prefix of the rerate-request file
    OutputDirectory         = pipeline_home/data/out/ood
    OutputPrefix            = ood_

  }
}
```

FCT_ExchangeRate

The FCT_ExchangeRate module converts the currency in the charge packets, discount packets, and tax packets to the home (system) currency or the customer's billing currency.

Dependencies

Requires a connection to the DAT_ExchangeRate module and the DAT_Currency module.

This module must run *after* the FCT_MainRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-65 lists the FCT_ExchangeRate registry entries.

Table 81-65 FCT_ExchangeRate Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
CurrencyDataModule	Specifies the connection to the DAT_Currency module. When currency module is specified, the resourceid field of the charge packet is updated to the new currency. The resourceid is needed only when loading the records into the BRM database by using Rated Event Loader.	No
ErrorMessages	Specifies whether error messages should be appended to the EDR container or should be suppressed.	Yes
ExchangeRateDataModule	Specifies the connection to the DAT_ExchangeRate module.	Yes
HomeCurrency	Specifies the local currency used by your company. The code must use three characters; for example, USD or DEM. You must set this value when the **RatingDateHome** registry value is set. This entry is used only if the **RatingDateHome** entry is used.	No
Mode	Specifies how to apply the exchange rate. Values are: • Normal - Converts the From Currency to the To Currency by multiplying the From Currency amount and exchange rate. • Reverse - Converts the To Currency to the From Currency by multiplying the To Currency amount and 1/exchange rate. Default mode is **Normal**.	No
RatingDateBilling	Specifies how to determine the validity date for the conversion from the rating currency to the billing currency. Values are: • SYSTEM • FILE • CDR • NONE Default = **NONE** **Note:** FCT_ExchangeRate module converts currency to home currency or billing currency. If you specify RatingDateBilling in addition to RatingDateHome and HomeCurrency parameters, it converts the currency to the home currency only.	No

Table 81-65 (Cont.) FCT_ExchangeRate Registry Entries

Entry	Description	Mandatory
RatingDateHome	Specifies how to determine the validity date for the conversion from the rating currency to the home currency. Values are: • SYSTEM • FILE • CDR • NONE Default = **NONE**	No

Sample Registry

```
ExchangRate
{
  ModuleName = FCT_ExchangeRate
  Module
  {
    Active = True
    ExchangeRateDataModule = ExchangeRateData
    RatingDateBilling  =  SYSTEM
    RatingDateHome = CDR
    HomeCurrency = DEM
    ErrorMessages = False
  }
}
```

Semaphore File Entries

Table 81-66 lists the FCT_ExchangeRate Semaphore file entries.

Table 81-66 FCT_ExchangeRate Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ExchangeRate.Module.Active = False
```

EDR Container Fields

Table 81-67 lists the FCT_ExchangeRate EDR container fields.

Table 81-67 FCT_ExchangeRate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CREATION_TIMESTAMP HEADER.CREATION_TIMESTAMP	Date	Read	Contains the EDR creation time stamp. This field is used if the exchange rate time (Home or Billing) is set to the file date.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. This field is used if the exchange rate time (Home or Billing) is set to the CDR date.
ASS_CBD DETAIL.ASS_CBD	Block-Index	Read	Block used for iteration over all associated charge breakdown records.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block-Index	Read	Block used for iteration over all charge packages.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	Integer	Read	Contains the record type. The billing and/or home currency is calculated for these record types: • 981 • 982 • 983 • 984 • 990 • 991 For record type 980, only the home currency is calculated.
CHARGE_CURRENCY_TYPE DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	Read/Write	Contains the currency type. The type **R** is read and the types **B** and **H** are calculated if specified.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Integer	Read/Write	Contains the charge amount that needs to be converted.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read/Write	Contains the amount in the charge currency for calculation.
INTERN_BILLING_CURRENCY DETAIL.ASS_CBD.CP.INTERN_BILLING_CURRENCY	String	Read	Contains the billing currency which is calculated and added with a charge package.

Table 81-67 (Cont.) FCT_ExchangeRate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_HOME_CURRENCY DETAIL.ASS_CBD.CP.INTERN_HOME_CURRENCY	String	Read	Contains the home currency which is calculated and added with an charge package. If no home currency is found the home currency from the registry is used for calculation.
EXCHANGERATE DETAIL.ASS_CBD.CP.EXCHANGERATE	String	Write	Contains the exchange rate recommended for TAP.
EXCHANGECURRENCY DETAIL.ASS_CBD.CP.EXCHANGECURRENCY	String	Read/Write	Contains the exchange currency recommended for TAP.
CHARGED_AMOUNT_VALUE_ORIG DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE_ORIG	Decimal	Write	Contains the charge amount in rating currency.
RESOURCE DETAIL.ASS_CBD.CP.RESOURCE	String	Write	Contains the balance element name
RESOURCE_ID DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Read/Write	Contains the balance element ID.
RESOURCE_ID_ORIG DETAIL.ASS_CBD.CP.RESOURCE_ID_ORIG	Integer	Read	Contains the rating balance element ID.
DISCOUNT_PACKET DETAIL.ASS_CBD.DP	Block	Read	Contains the discount related information for the event.
DISCOUNT_PACKET_GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Decimal	Read/Write	Contains the discounted amount.
DISCOUNT_PACKET_GRANTED_AMOUNT_ORIG DETAIL.ASS_CBD.DP.GRANTED_AMOUNT_ORIG	Decimal	Write	Contains the discounted amount in the initial rated currency.
DISCOUNT_PACKET_INTERN_SRC_PACKET_INDEX DETAIL.ASS_CBD.DP.INTERN_SRC_PACKET_INDEX	Integer	Read	Contains the charge packet number for which this discount is given.
DISCOUNT_PACKET_RESOURCE_ID DETAIL.ASS_CBD.DP.RESOURCE_ID	Integer	Read/Write	Contains the balance element ID.
DISCOUNT_PACKET_RESOURCE_ID_ORIG DETAIL.ASS_CBD.DP.RESOURCE_ID_ORIG	Integer	Write	Contains the currency for which discount has been given originally.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Contains the block creation number. These numbers (980, 981, 982, 983, 984, 990, 991) are only considered for exchange rate. Any other number of ASS_CBD will be ignored.

Table 81-67 (Cont.) FCT_ExchangeRate EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_CHARGE_PACKET_TAX_PACKET DETAIL.ASS_CBD.TP	Block	Read	Contains tax related information for the event.
CHARGED_INFO_ID DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Integer	Read	Contains charge packet index corresponding to the tax.
TP_TAX_VALUE DETAIL.ASS_CBD.TP.TAX_VALUE	Decimal	Read/Write	Contains the tax amount
TP_TAX_VALUE_ORIG DETAIL.ASS_CBD.TP.TAX_VALUE_ORIG	Decimal	Write	Contains the tax amount in the originally rated currency.
CUST_A_CURRENCY DETAIL.CUST_A.CURRENCY	String	Read	Contains the billing currency of the customer.

FCT_Filter_Set

The FCT_Filter_Set module determines whether an event data record (EDR) is eligible for the system charge offers and system discounts contained in a filter set. If so, it adds those system charge offers and discounts to a customer's list of purchased products.

Dependencies

This module must run *after* the FCT_Account module.

FCT_Filter_Set requires the following connections:

- BRM database
- DAT_Discount module
- DAT_AccountBatch module

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-68 lists the FCT_Filter_Set registry entries.

Table 81-68 FCT_Filter_Set Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountBatch module.	Yes
Active	Specifies if the module is active or inactive. • **True** = Active • **False** = Inactive **Note:** When the module is active, it takes over the function of applying system discounts from the FCT_DiscountAnalysis module.	Yes
DiscountDataModule	Specifies the connection to the DAT_Discount module.	Yes

Table 81-68 (Cont.) FCT_Filter_Set Registry Entries

Entry	Description	Mandatory
InfranetConnection	Specifies the connection to the BRM database.	Yes

Sample Registry

```
#------------------------
# Segment FCT
#------------------------
Segment
{
    ModuleName = FCT_Filter_Set
    Module
    {
        Active = True
        DiscountDataModule = ifw.DataPool.DiscountModelDataModule
        AccountDataModule = ifw.DataPool.CustomerData
        InfranetConnection = ifw.DataPool.LoginInfranet
    }
}
```

Semaphore File Entries

Table 81-69 lists the FCT_Filter_Set Semaphore file entries.

Table 81-69 FCT_Filter_Set Semaphore File Entry

Entry	Description
Reload	Reloads data from the database into FCT_Filter_Set.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Segment.Module.Reload{}
```

EDR Container Fields

This module can read any valid EDR fields (defined in **container.dsc**) for matching criteria (EDR field name and value).

> **Note:**
>
> EDR field names read by this module cannot exceed a depth of 6 levels. Exceeding this limit prevents the pipeline from starting and results in an error message.

FCT_FirstUsageNotify

The FCT_FirstUsageNotify module sets the batch rating output stream for charge offers and discounts that start on first usage and sets an error code in the EDR for suspending events that use those charge offers and discounts while their validity periods are being set.

For more information, see "About Suspending EDRs for Products and Discounts that Start on First Usage".

> **Note:**
>
> To process first-usage events in the real-time rerating pipeline, use the "ISC_FirstProductRealtime" iScript.

Dependencies

Run this module before the FCT_ApplyBalance and FCT_Reject modules.

Requires a connection to DAT_AccountBatch and FCT_Reject.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-70 lists the FCT_FirstUsageNotify registry entries.

Table 81-70 FCT_FirstUsageNotify Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive. • **True** = Active • **False** = Inactive You can use this entry in a semaphore file.	Yes
CustomerDataModule	Specifies a connection to the DAT_AccountBatch module.	Yes
FirstUsageNotifyOutput	Specifies the output stream for the list of charge offers and discounts used that are set to start on first usage. Default = **FirstUsageNotifyOutput**	Yes
RejectModule	Specifies a connection to the FCT_Reject module. FCT_FirstUsageNotify uses FCT_Reject to determine whether an EDR will be rejected for reasons other than setting the validity. FCT_FirstUsageNotify does not set validity if the EDR will be rejected.	Yes

Sample Registry

```
FirstUsageNotify
{
    ModuleName = FCT_FirstUsageNotify
    Module
    {
```

```
            Active = True
            CustomerDataModule = ifw.DataPool.Account
            RejectModule = ifw.Pipelines.MyPipeline1.Reject
            FirstUsageNotifyOutput = FirstUsageNotifyOutput
        }
    }
```

Semaphore File Entries

Table 81-71 lists the FCT_FirstUsageNotify Semaphore file entry.

Table 81-71 FCT_FirstUsageNotify Semaphore File Entry

Entry	Description
Active	Specifies whether the module is active or inactive. • **True** = Active • **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.FirstUsageNotify.Module.Active = False
```

EDR Container Fields

Table 81-72 lists the FCT_FirstUsageNotify EDR container fields.

Table 81-72 FCT_FirstUsageNotify EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TRANSACTION_ID INTERNAL.TRANSACTION_ID	String	Read	Specifies the transaction ID.
CUST_A_INTERN_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Read	Contains an index of the customer's purchased charge offers that were used for rating.
CUST_A_ACCOUNT_PARENT_ID DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Specifies the customer account POID.
CUST_A_PRODUCT_ID DETAIL.CUST_A.PRODUCT.OFFERING_POID	String	Read	Specifies the POID of the account's charge offer used to rate the event.
CUST_A_PRODUCT_FIRST_USAGE_INDICATOR DETAIL.CUST_A.PRODUCT.FIRST_USAGE_INDICATOR	String	Read	Specifies whether the charge offer is configured to start when first used and the first-usage validity period status.
FU_DISCOUNT_OBJECTS DETAIL.ASS_CBD.FU_DISCOUNT_OBJECTS	String	Read	Specifies the account's first-usage discounts that were applied to the event.

Table 81-72 (Cont.) FCT_FirstUsageNotify EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
RECYCLE_KEY DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY	String	Write	Specifies the recycle key. If the charge offer or discount starts on first usage and its validity period is not set, this field is set to **FirstUsageValidity**.
UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Specifies the difference between local time and UTC time.
ASSOCIATED_INFRANET_BILLING DETAIL.ASS_PIN	String	Write	The Associated BRM Billing record.
FIRST_USAGE_SET_VALIDITY DETAIL.ASS_PIN.FIRST_USAGE	String	Write	The First-usage data block.
FIRST_USAGE_ACCOUNT_POID DETAIL.ASS_PIN.FIRST_USAGE.ACCOUNT_POID_STR	String	Write	Specifies the customer account POID.
FIRST_USAGE_OFFERING_POID DETAIL.ASS_PIN.FIRST_USAGE.OFFERING_POID_STR	String	Write	Specifies the POID of the account's charge offer or discount used to rate or discount the event.
FIRST_USAGE_START_TIME DETAIL.ASS_PIN.FIRST_USAGE.START_T	String	Write	Specifies the validity period start time, which is based on the event start time.
FU_UTC_TIME_OFFSET DETAIL.ASS_PIN.FIRST_USAGE.UTC_TIME_OFFSET	String	Read	Specifies the difference between first-usage start time and UTC time.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	String	Read	Specifies the EDR start timestamp.
CUST_A_PRODUCT_SERVICE_ID DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Specifies the POID of the service object
CUST_A_PRODUCT_SERVICE_TYPE DETAIL.CUST_A.PRODUCT.SERVICE_TYPE	String	Read	Specifies the POID type of the service object
CUST_A_PRODUCT_SERVICE_TYPE DETAIL.ASS_PIN.FIRST_USAGE.SERVICE_POID_STR	String	Read	Specifies the POID of the service object which has FU charge offer uninitialized

FCT_GlobalRating

The FCT_GlobalRating module rates all EDRs with a default set of charges. See "About Global Rating".

Dependencies

This module must run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-73 lists the FCT_GlobalRating registry entries.

Table 81-73 FCT_GlobalRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
EdrRatePlans	Specifies a set of charges that are used for rating. You can use this entry in a semaphore file.	Yes

Sample Registry

```
GlobalRating
{
  ModuleName = FCT_GlobalRating
  Module
  {
    Active = True
    EdrRatePlans
    {
      RatePlan_1
      RatePlan_2
      RatePlan_3
    }
  }
}
```

Semaphore File Entries

Table 81-74 lists the FCT_GlobalRating Semaphore file entries.

Table 81-74 FCT_GlobalRating Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrRatePlans	Specifies a set of charges that are used for rating.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.GlobalRating.Module.Active = True
```

EDR Container Fields

Table 81-75 lists the FCT_GlobalRating EDR container fields.

Table 81-75 FCT_GlobalRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Create	Block to hold the rating data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	Create	Block created for each **EdrRatePlans** entry.
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the EDR start time stamp. This value is used in the charge packet.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the A number for zoning. This value is used in the charge packet.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number for zoning. This value is used in the charge packet.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type. This value is always set to **980**.
CHARGE_TYPE DETAIL.ASS_CBD.CP.CHARGE_TYPE	String	Write	Contains the charge type. This value is always set to **N**.
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging start timestamp from the DETAIL field.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	Contains the charge code that is used for zoning and rating.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Write	Contains the A number zoning information used by the FCT_PreRating module. See "FCT_PreRating".
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Write	Contains the B number zoning information used by the FCT_PreRating module. See "FCT_PreRating".

FCT_IRules

The FCT_IRules module evaluates iRules. You use iRules to perform functions such as mapping EDR data fields and splitting EDR containers to different output streams. You group rules together in a rule set.

You can store rules in the database or in an ASCII file.

> **Note:**
>
> FCT_IRules cannot read files written in XML format. You can, however, use the **irules2db.pl** script to load iRules written in XML into the database.

Dependencies

If the data is read from the database, this module requires a connection to the Pipeline Manager database.

This module can run anywhere, depending on the data that is being processed.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-76 lists the FCT_IRules registry entries.

Table 81-76 FCT_IRules Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database if rules are stored in the database.	Yes
Descriptions	Specifies the rule set descriptions. The rule sets are evaluated in the specified order. You can use this entry in a semaphore file.	No
Rules	Specifies the rule sets that the module evaluates. You can use this entry in a semaphore file.	Yes, when using the database interface. No, when using a file interface.
Source	Specifies the source of the rules: • **File** • **Database**	Yes

Sample Registry Entry for the Database Interface

```
IRules
{
    ModuleName = FCT_IRules
    Module
    {
        Active = TRUE
        Source = Database
        DataConnection = ifw.DataPool.DataConnection
        Rules
        {
            CIBER_VAL
        }
    }
}
```

Sample Registry Entry for the File Interface

```
Router
{
    ModuleName = FCT_IRules
    Module
    {
        Active = TRUE
        Source = File
        Rules
        {
        }
        Descriptions
        {
            ServiceRequestRouter = ./iScriptLib/AAA/IRL_Router.irl
        }
    }
}
```

Semaphore File Entries

Table 81-77 lists the FCT_IRules Semaphore file entries.

Table 81-77 FCT_IRules Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Descriptions	Specifies the rule set descriptions. The rule sets are evaluated in the specified order.
Reload	Reloads rules from the database or an ASCII file.
Rules	Specifies the rule sets that the module evaluates.

Sample Semaphore File Entry

```
ifw.Pipelines.PRE_RECYCLE.Functions.PreProcessing.FunctionPool.PipelineSplit.Module.Reload {}
```

EDR Container Fields

The EDR container fields that are accessed by FCT_IRule are those that you specify in the rules.

Database Interface

FCT_IRules uses the following database tables to store the generic rules:

- **IFW_RULESET**. Specifies a rule set for linking a related set of rules.
- **IFW_RULESETLIST**. Links a rule set with its rules. Each rule has a rank that specifies the order in which it is evaluated.
- **IFW_RULE**. Stores the iRules.
- **IFW_RULEITEM**. Contains the conditions, the result and the rank for a rule item.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

File Interface

You can store iRules in an ASCII file by using a syntax that is similar to XML:

- Each tag must start on a separate line.
- Blank lines are allowed.
- Comment lines must start with a pound symbol: #

The rules and rule items are ranked by their order in the file; the first having the highest rank.

Loading Rule Sets from the Database

You can load rule sets from the Pipeline Manager database. iRule components are stored in the following tables:

- **IFW_RULESET**
- **IFW_RULESETLIST**
- **IFW_RULE**
- **IFW_RULEITEM**

Loading Rule Sets from an ASCII File

The following example file shows the syntax of a rule set:

```
###############################################################################
# Example ruleset file
###############################################################################
<RULESET>
#==============================================================================
```

```
# The first rule of this ruleset
#===============================================================================
<RULE MapRule1>
<INIT_SCRIPT>
String code = edrString(DETAIL.SERVICE_CODE) + "_GK2";
</INIT_SCRIPT>
#-------------------------------------------------------------------------------
# The first ruleitem of this rule
#-------------------------------------------------------------------------------
<RULEITEM discount>
<CONDITION>
/* The text between <CONDITION> and </CONDITON> is directly passed to the
* the interpreter. Lines with # are no comments here!
*/
edrString(DETAIL.RECORD_TYPE ) =~ "02*"; // this is a pattern compare rule
edrLong(DETAIL.QUANTITY) > 30; // this is a normal condition
</CONDITION>
<RESULT>
/* This iScript is executed if the conditions specified in the <CONDITION>
* tag before are matching the current edr container */
edrDecimal(DETAIL.CHARGED_AMOUNT) = 2.50; // set the amount to DM 2.50
</RESULT>
</RULEITEM>
#-------------------------------------------------------------------------------
# Here can be specified some more RULEITEMS
#-------------------------------------------------------------------------------
</RULE>
#===============================================================================
# The second rule of this ruleset
#===============================================================================
<RULE MapRule2>
#-------------------------------------------------------------------------------
# Put the ruleitems for MapRule2 here
#-------------------------------------------------------------------------------
</RULE>
</RULESET>
```

FCT_IScript

The FCT_IScript module runs iScripts. The scripts are run in the order specified in the registry.

Dependencies

If the iScripts are stored in the database, this module requires a connection to the Pipeline Manager database.

This module can run anywhere, depending on the data that is being processed.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-78 lists the FCT_IScript registry entries.

Table 81-78 FCT_IScript Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies a connection to the Pipeline Manager database.	Yes, if the module gets data from the database.
Scripts	Specifies the scripts to run. • If the scripts are stored in the database, each value in the entry specifies the SCRIPTCODE field in the INT_ISCRIPT database table. • If the scripts are stored in a file, each section in the file must have a registry entry FileName = *file*. **Note:** The section for each script can store a number of entries that are passed as global constants to the interpreter.	Yes
Scripts.*ScriptName*	The name of the script as used in the registry.	N/A
Scripts.*ScriptName*.**FileName**	The name and path of the script.	N/A
Scripts.*ScriptName.Registry_Parameter*	Registry parameters used in the iScript. The example below uses the GL_Code parameter: GL_CODE = 1514	N/A
Source	Specifies the source of the iScripts. • **File** • **Database**	Yes

Sample Registry for the File Interface

```
ConsolidatedCP
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         ConsolidatedCPIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ConsolidatedCP.isc
            GL_CODE = 1514
         }
      }
   }
}
```

Sample Registry for the Database Interface

```
IScript
{
```

```
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = Database
    DataConnection = ifw.DataPool.Login
    Scripts
    {
      Mapping
      {
        # can be used as reg.Arg1 in the IScript
        Arg1 = any_argument_value
        }
        Specials
        {
      }
    }
  }
}
```

Semaphore File Entries

Table 81-79 lists the FCT_IScript Semaphore file entries.

Table 81-79 FCT_IScript Semaphore File Entries

Entry	Description
Active	Activates or deactivates the module. **TRUE** = Activate. **FALSE** = Deactivate.
Reload	Reloads iScripts from the database or an ASCII file.
Scripts	Section with scripts to run. In case of Source = DATABASE each value in the section specifies the SCRIPTCODE of a script in database table INT_ISCRIPT. In case of Source = FILE each section within the section must have a registry entry FileName = *file*.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Standard.FunctionPool.IScript.Module.Reload {}
```

Database Tables

The FCT_IScript module uses the IFW_ISCRIPT database table to store iScripts.

File Interface

The iScript source code is stored in a simple ASCII file. The file is loaded and compiled at startup.

FCT_ItemAssign

The FCT_ItemAssign module assigns bill items to events.

Dependencies

FCT_ItemAssign requires a connection to the DAT_ItemAssign module.

Must run after the FCT_Account, rating, and discounting modules and before the FCT_BillingRecord module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-80 lists the FCT_ItemAssign registry entries.

Table 81-80 FCT_ItemAssign Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
DataModule	Specifies the connection to the DAT_ItemAssign module.	Yes

Sample Registry

```
Item Assignment
{
  ModuleName = FCT_ItemAssign
  Module
  {
    Active = True
    DataModule = ifw.DataPool.ItemAssignDataModule
  }
}
```

EDR Container Fields

The FCT_ItemAssign module uses the EDR container fields listed in Table 81-80:

Table 81-81 FCT_ItemAssign EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.ITEM_TAG	String	Read	Contains the item tag.
DETAIL.CUST_A. PRODUCT.SERVICE_USED_ITEM_POID	String	Write	Contains the item POID for the event.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Contains the charge offer ID of the charge offer used for rating.
DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Contains the customer's account number.
DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Contains the ID of the charge offer.

Table 81-81 (Cont.) FCT_ItemAssign EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC time zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/- HHMM.
DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the start timestamp of the event. The time zone information for this timestamp is stored in the BDR_UTC_TIME_OFFSET field.

FCT_MainRating

The FCT_MainRating module performs the main rating functionality in a pipeline. See "About Main Rating".

Dependencies

The FCT_MainRating module requires a connection to the following data modules:

- DAT_Calendar
- DAT_Currency
- DAT_TimeModel
- DAT_Rateplan
- DAT_PriceModel
- DAT_ModelSelector

This module must run after at least one of the following modules:

- FCT_CarrierIcRating
- FCT_CustomerRating
- FCT_GlobalRating

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-82 lists the FCT_MainRating registry entries.

Table 81-82 FCT_MainRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
CalendarDataModule	Specifies the connection to the DAT_Calendar module.	Yes
CurrencyDataModule	Specifies the connection to the DAT_Currency module.	No
ModelSelectorDataModule	Specifies the connection to the DAT_ModelSelector module.	Yes
PriceDataModule	Specifies the connection to the DAT_PriceModel module.	Yes
RateplanDataModule	Specifies the connection to the DAT_Rateplan module.	Yes
TimeDataModule	Specifies the connection to the DAT_TimeModel module.	Yes

Sample Registry

```
MainRating
{
  ModuleStart = FCT_MainRating
  Module
  {
    Active = True
    RateplanDataModule = RateplanDataModule
    CalendarDataModule = CalendarDataModule
    TimeDataModule = TimeDataModule
    PriceDataModule = PriceDataModule
    CurrencyDataModule = ifw.DataPool.CurrencyDataModule
    ModelSelectorDataModule = Model selector module
  }
}
```

Semaphore File Entries

Table 81-83 lists the FCT_MainRating Semaphore file entry.

Table 81-83 FCT_MainRating Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.MainRating.Module.Active = False
```

EDR Container Fields

Table 81-84 lists the FCT_MainRating EDR container fields.

Table 81-84 FCT_MainRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block	Read	Data block for rate data.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block	Read	Charge packet for rate data.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Read	Contains the wholesale charge amount value.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Contains the record number.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read/Write	Contains the charge code.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Write	Contains the charge type.
TIMEMODEL_CODE DETAIL.ASS_CBD.CP.TIMEMODEL_CODE	String	Write	Contains the time model code.
TIMEZONE_CODE DETAIL.ASS_CBD.CP.TIMEZONE_CODE	String	Write	Contains the time zone code.
DAY_CODE DETAIL.ASS_CBD.CP.DAY_CODE	String	Write	Contains the special day code.
TIME_INTERVAL_CODE DETAIL.ASS_CBD.CP.TIME_INTERVAL_CODE	String	Write	Contains the time interval code.
PRICEMODEL_CODE DETAIL.ASS_CBD.CP.PRICEMODEL_CODE	String	Write	Contains the pricing code.
PRICEMODEL_TYPE DETAIL.ASS_CBD.CP.PRICEMODEL_TYPE	String	Write	Contains the pricing type.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Write	Contains the service code.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Write	Contains the service class.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Write	Contains the charge amount value for the event.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Write	Contains the currency amount.
CHARGED_CURRENCY_TYPE DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	Write	Contains the currency type.
CHARGED_TAX_TREATMENT DETAIL.ASS_CBD.CP.CHARGED_TAX_TREATMENT	String	Write	Contains the tax treatment.

Table 81-84 (Cont.) FCT_MainRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_CHARGING_START_TIMESTAMP DETAIL.ASS_CBD.CP.CHARGING_START_TIMESTAMP	Date	Write	Contains the charging time stamp.
ROUNDED_QUANTITY_VALUE DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_VALUE	Decimal	Write	Contains the rounded quantity value.
ROUNDED_QUANTITY_UOM DETAIL.ASS_CBD.CP.ROUNDED_QUANTITY_UOM	String	Write	Contains the rounded quantity UoM.
USAGE_GL_ACCOUNT_CODE DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Write	Contains the G/L code.
REVENUE_GROUP_CODE DETAIL.ASS_CBD.CP.REVENUE_GROUP_CODE	String	Write	Contains the revenue group.
RUMGROUP DETAIL.ASS_CBD.CP.RUMGROUP	String	Write	Contains the RUM group.
RUM DETAIL.ASS_CBD.CP.RUM	String	Write	Contains the RUM.
RESSOURCE DETAIL.ASS_CBD.CP.RESOURCE	String	Write	Contains the balance element.
INTERN_DISCOUNT_MODEL DETAIL.ASS_CBD.CP.INTERN_DISCOUNT_MODEL	Integer	Write	Contains the discount.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Write	Contains the internal charge.
INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Integer	Write	Contains the charge version.
INTERN_FIX_COST DETAIL.ASS_CBD.CP.INTERN_FIX_COST	Decimal	Read	Contains the fixed cost amount.
INTERN_PRICE_MDL_STEP_INFO DETAIL.ASS_CBD.CP.INTERN_PRICE_MDL_STEP_INFO	String	Write	Contains information about the pricing steps used to calculate the charge in the charge packet.
DETAIL.CUST_A.PRODUCT.RATEPLAN_CODE	String	Read	Contains the charge code of the charge offer to match with the charge in the charge breakdown record.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Read	Contains the rating charge offer indexes. This is a comma-separated list of all rating charge offers' indexes associated with the same service and event, and their priorities.

Table 81-84 (Cont.) FCT_MainRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Write	Contains the index of the highest priority rating charge offer. This is the charge offer with the highest rate priority for an event.
DETAIL.CUST_A.LEAST_COST	Integer	Read	Indicates whether to use least cost rating for an EDR. **1** turns it off; **2** turns it on.
DETAIL.CUST_A.PROMOTIONAL_SAVING	Integer	Read	Indicates whether to use promotional savings for an EDR. **1** turns it off; **2** turns it on.
DETAIL.CUST_A.PROMOTION	Integer	Read	Indicates whether to use overlay promotions for an EDR. **1** turns it off; **2** turns it on.

FCT_MainZoning

The FCT_MainZoning module performs zoning for multi-segment zoning.

Dependencies

Requires a connection to the DAT_Zone module.

This module must run after the FCT_SegZoneNoCust module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-85 lists the FCT_MainZoning registry entries.

Table 81-85 FCT_MainZoning Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
ZoneDataModule	Specifies the connection to the DAT_Zone module.	Yes

Sample Registry

```
MainZoning
{
  ModuleName = FCT_MainZoning
  Module
  {
    Active = True
    ZoneDataModule = integRate.DataPool.ZoneDataModule
  }
}
```

Semaphore File Entries

Table 81-86 lists the FCT_MainZoning Semaphore file entries.

Table 81-86 FCT_MainZoning Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.MainZoning.Module.Active = False
```

EDR Container Fields

Table 81-87 lists the FCT_MainZoning EDR container fields.

Table 81-87 FCT_MainZoning EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the zone for the A number.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the zone for the B number.
ASS_ZBD_ZONE_DESCRIPTION DETAIL.ASS_ZBD.ZP.ZONE_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
ASS_ZBD_RECORD_NUMBER DETAIL.ASS_ZBD.RECORD_NUMBER	Integer	Read	Contains the record number.

Table 81-87 (Cont.) FCT_MainZoning EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_ZBD_ZONE_RESULT_VALUE_WS DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_WS	String	Write	Contains the wholesale zone.
ASS_ZBD_ZONE_RESULT_VALUE_RT DETAIL.ASS_ZBD.ZP.ZONE_RESULT_VALUE_RT	String	Write	Contains the retail zone.
ASS_ZBD_ZONEMODEL_CODE DETAIL.ASS_ZBD.ZP.ZONEMODEL_CODE	String	Write	Contains the zone model code.
ASS_ZBD_INTERN_ZONE_MODEL DETAIL.ASS_ZBD.ZP.INTERN_ZONE_MODEL	Integer	Read	Contains the zone model.
ASS_ZBD_INTERN_APN_GROUP DETAIL.ASS_ZBD.ZP.INTERN_APN_GROUP	String	Write	Contains the APN group.
ASS_ZBD_ZONE_ENTRY_NAME DETAIL.ASS_ZBD.ZP.ZONE_ENTRY_NAME	String	Write	Contains the destination description for this combination of service code and impact category.

FCT_NOSP

The FCT_NOSP module maps network source and destination to new values. See "Identifying the Network Operator/Service Provider ".

Dependencies

Requires a connection to the DAT_NOSP module.

This module must be run before segment rating is performed.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-88 lists the FCT_NOSP registry entries.

Table 81-88 FCT_NOSP Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_NOSP module. See "DAT_NOSP".	Yes
MapGroup	Specifies the map group that the NOSP mappings belong to. You can use this entry in a semaphore file.	Yes

Sample Registry

```
Zoning
{
  ModuleName = FCT_NOSP
  Module
  {
    Active = True
    DataModule = ifw.DataPool.NospData
    MapGroup = MOBILE
  }
}
```

Semaphore File Entries

Table 81-89 lists the FCT_NOSP Semaphore file entries.

Table 81-89 FCT_NOSP Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
MapGroup	Specifies the map group.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.NOSP.Module.Active = False
```

EDR Container Fields

Table 81-90 lists the FCT_NOSP EDR container fields.

Table 81-90 FCT_NOSP EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read/Write	Contains the source network.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read/Write	Contains the destination network.
A_NUMBER DETAIL.A_NUMBER	String	Read	Contains the A number.

FCT_NumberPortability

The FCT_NumberPortability module specifies the new network operator for an existing phone number.

Dependencies

Requires a connection to the DAT_NumberPortability module.

This module must be run before the zoning and rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-91 lists the FCT_NumberPortability registry entries.

Table 81-91 FCT_NumberPortability Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive Default = **False**. You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_NumberPortability module.	Yes
DefaultSourceNetwork	Specifies the default network operator ID of the source network. This ID is used if there is no ID present in the data retrieved from the DAT_NumberPortability module.	Yes
DefaultDestinationNetwork	Specifies the default network operator ID of the destination network. This ID is used if there is no ID present in the data retrieved from the DAT_NumberPortability module.	Yes
OverwriteNetwork	If the **DefaultSourceNetwork** and **DefaultDestinationNetwork** fields are empty, overwrites the source and destination network with the value configured in the DAT_NumberPortability module. Default = **True**.	No
OverwriteNetworkType	If the SOURCE_NETWORK_TYPE and DESTINATION_NETWORK_TYPE fields are empty, overwrites the SOURCE_NETWORK_TYPE and DESTINATION_NETWORK_TYPE fields with the type of network that is populated in the source and destination network. Default = **True**.	No

Sample Registry

```
NumberPortability
{
  ModuleName = FCT_NumberPortability
  Module
  {
    Active = True
    DataModule = integrate.DataPool.NPortData
    DefaultSourceNetwork = D030
    DefaultDestinationNetwork = D017
  }
}
```

Semaphore File Entries

Table 81-92 lists the FCT_NumberPortability Semaphore file entry.

Table 81-92 FCT_NumberPortability Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.NumberPortability.
Module.Active = False
```

EDR Container Fields

Table 81-93 lists the FCT_NumberPortability EDR container fields.

Table 81-93 FCT_NumberPortability EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
A_NUMBER DETAIL.A_NUMBER	String	Read	Specifies the event originator.
B_NUMBER DETAIL.B_NUMBER	String	Read	Specifies the event receiver.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Specifies the charging timestamp. The format is: *YYYYMMDDhhmmss*
IGNORE_NP DETAIL.IGNORE_NP	Integer	Read/ Write	Specifies whether FCT_NumberPortability should look for network operator IDs for A and B number. Default is **0** (cleared).
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Write	Specifies the source network. This can either be the PLMN ID or any logical operator code.
SOURCE_NETWORK_TYPE DETAIL.SOURCE_NETWORK_TYPE	String	Write	Specifies the source network type, for example GSM 900.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Write	Specifies the network to which an event is routed.

Table 81-93 (Cont.) FCT_NumberPortability EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
DESTINATION_NETWORK_TYPE DETAIL.DESTINATION_NETWORK_TYPE	String	Write	Specifies the destination network type, for example GSM 900.

FCT_Opcode

The FCT_Opcode module uses the DAT_ConnectionPool module to connect to the CM and calls the appropriate opcode for the request.

Dependencies

The FCT_Opcode module requires a connection to the "DAT_ConnectionPool" module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-94 lists the FCT_Opcode registry entries.

Table 81-94 FCT_Opcode Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive **True** = Active (default) **False** = Inactive	Yes
Retries	Specifies the number of times to try the request on the CM Default = **2**	Yes
Logging	Logs each opcode called from the processing pipeline Default = **False**	Yes
ConnectionPoolDataModule	Specifies a connection to the DAT_ConnectionPool module.	Yes

Sample Registry

```
EdrOpcodeCall
{
  ModuleName = FCT_Opcode
  Module
  {
    Active = True
    Retries = 2
    Logging = True
    ConnectionPoolDataModule = ifw.DataPool.CMConnectionPool.Module
  }
}
```

EDR Container Fields

FCT_Opcode uses the EDR container fields listed in Table 81-95:

Table 81-95 FCT_Opcode Container Fields

Alias Field Name Default Field Name	Type	Access	Description
OPCODE_FLAG DETAIL.OPCODE_FLAG	Integer	Read	The flag that specifies the behavior of the opcode
OPCODE_NODE DETAIL.OPCODE_NODE	String	Read	Name of the opcode
PCM_OP_EBUF DETAIL.PCM_OP_EBUF	pin_ebuf_t	Read	Error buffer

FCT_PrefixDesc

The FCT_PrefixDesc module maps phone number prefixes to destination descriptions. See "Creating Call Destination Descriptions".

Dependencies

Requires a connection to the DAT_PrefixDesc module.

This module can run from anywhere.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-96 lists the FCT_PrefixDesc registry entries.

Table 81-96 FCT_PrefixDesc Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
PrefixDataModule	Specifies the connection to the DAT_PrefixDesc module.	Yes

Sample Registry

```
{
  ModuleName = FCT_PrefixDesc
  Module
  {
    Active = True
    PrefixDataModule = PrefixDescData
```

```
    }
}
```

Semaphore File Entries

Table 81-97 lists the FCT_PrefixDesc Semaphore file entry.

Table 81-97 FCT_PrefixDesc Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PrefixDesc.Module.Active = False
```

EDR Container Fields

Table 81-98 lists the FCT_PrefixDesc EDR container fields.

Table 81-98 FCT_PrefixDesc EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
B_NUMBER DETAIL.B_NUMBER	String	Read	Contains the B number.
DESCRIPTION DETAIL.DESCRIPTION	String	Write	Contains the call destination description.

FCT_PreRating

The FCT_PreRating module calculates zones and creates impact category.

Dependencies

Requires a connection to the DAT_Rateplan and the DAT_Zone module.

This module must be run before the FCT_MainRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-99 lists the FCT_PreRating registry entries.

Table 81-99 FCT_PreRating Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
RateplanDataModule	Specifies the connection to the DAT_Rateplan module.	Yes
ZoneDataModule	Specifies the connection to the DAT_Zone module.	Yes

Sample Startup Registry

```
PreRating
{
  ModuleStart = FCT_PreRating
  Module
  {
    Active = True
    RateplanDataModule = ifw.DataPool.RateplanDataModule
    ZoneDataModule = ifw.DataPool.ZoneDataModule
  }
}
```

Semaphore File Entries

Table 81-100 lists the FCT_PreRating Semaphore file entry.

Table 81-100 FCT_PreRating Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreRatingZone.Module.Active = False
```

EDR Container Fields

Table 81-101 lists the FCT_PreRating EDR container fields.

Table 81-101 FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp. The time stamp is used for calculating the impact category by comparing it to the dates in the VALID_FROM and VALID_TO fields in the **IFW_STANDARD_ZONE** database table.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the service code. The service code is used for calculating the impact category by comparing it with the value in the SERVICECODE field in the **IFW_STANDARD_ZONE** database table. The module writes the service code to the ASS_CBD_SERVICE_CODE and SERVICE_CLASS_USED fields.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the service class. The module writes the service class to the ASS_CBD_SERVICE_CODE and SERVICE_CLASS_USED fields.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	String	Read	Contains the charge breakdown record number. Charge breakdown records are processed only if the record number = 0 (newly created).
ASS_CBD_SERVICE_CODE DETAIL.ASS_CBD.SERVICE_CODE	String	Write	Contains the service code. Set with the value of the INTERN_SERVICE_CODE field.
ASS_CBD_ZONEMODEL_CODE DETAIL.ASS_CBD.CP.ZONEMODEL_CODE	String	Write	Contains the zone model code. Set with the value of the INTERN_SERVICE_CLASS field.

Table 81-101 (Cont.) FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Write	Contains the impact category. Set with the zoning results by using the value from either the ZONE_WS or ZONE_RT in the **IFW_STANDARD_ZONE** database table, depending on charge type.
RATEPLAN_CODE DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Write	Contains a comma-separated list of charge codes for all rating charge offers. This list arranged by charge offer priority, with the highest priority first and the lowest priority last. Set with the value of the CODE field in the **IFW_RATEPLAN** database table.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Write	Contains the charge type. Set with the value of the TYPE field in the **IFW_RATEPLAN** database table.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Write	Contains the service code. Set with the value from the INTERN_SERVICE_CODE field.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Write	Contains the service class. Set with the value from the INTERN_SERVICE_CLAS S field.
INTERN_ORIGIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_ORIGIN_NUM_ZONE	String	Read	Contains the area code for the A number. The area code is used for calculating the impact category by comparing it to ORIGIN_AREACODE field in the **IFW_STANDARD_ZONE** database table.
INTERN_DESTIN_NUM_ZONE DETAIL.ASS_CBD.CP.INTERN_DESTIN_NUM_ZONE	String	Read	Contains the area code for the B number. The area code is used for calculating the impact category by comparing it to DESTIN_AREACODE field in the **IFW_STANDARD_ZONE** database table.

Table 81-101 (Cont.) FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the charge. Set with the value from the RATEPLAN field in the **IFW_RATEPLAN** database table.
INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Integer	Write	Contains the charge version. Set with the value from the VERSION field in the **IFW_RATEPLAN_VER** database table.
ASS_CBD_INTERN_ZONE_MODEL DETAIL.ASS_CBD.CP.INTERN_ZONE_MODEL	Integer	Write	Contains the zone model. Set with the value from the ZONEMODEL field in the **IFW_RATEPLAN_VER** database table.
ASS_CBD_INTERN_APN_GROUP DETAIL.ASS_CBD.CP.INTERN_APN_GROUP	String	Write	Contains the APN group. Set with the value from the APN_GROUP field in the **IFW_ZONEMODEL** database table.
ASS_CBD_INTERN_GEOMODEL DETAIL.ASS_CBD.CP.INTERN_GEOMODEL	Integer	Write	Contains the geographical model, if the MODELTYPE field in the **IFW_ZONEMODEL** database table is set to **L**. Set with the value from the GEOMODEL field in the **IFW_ZONEMODEL** database table.
ASS_CBD_INTERN_RULESET DETAIL.ASS_CBD.CP.INTERN_RULESET	String	Write	Contains the rule set, if the MODELTYPE field in the **IFW_ZONEMODEL** database table is set to **L**. Set with the value from the RULESET field in the **IFW_GEOGRAPHICAL_MODEL** database table.
ASS_CBD_ZONE_DESCRIPTION DETAIL.ASS_CBD.CP.ZONE_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
ASS_ZBD_ZONE_ENTRY_NAME DETAIL.ASS_ZBD.ZP.ZONE_ENTRY_NAME	String	Write	Contains the destination description for displaying on invoices.
RATE_PLAN_NAME DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Write	Contains the charge name for the purchased charge offer.
INTERN_RATING_PRODUCTS DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Write	Contains the indexes of the candidate charge offers that can be used for rating.

Table 81-101 (Cont.) FCT_PreRating EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_FOUND_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Write	Contains the purchased charge offer index of the charge offer or service used.
CUST_A_LEAST_COST_RATING DETAIL.CUST_A.LEAST_COST	String	Write	Specifies if least cost rating is to be used for rating the EDR.

FCT_PreRecycle

The FCT_PreRecycle module gets the file of rejected EDRs from the reject stream output directory. The module puts the reject EDR file into the pipeline input directory for recycling. It uses the same input folder as the incoming CDR files.

See:

- Configuring Standard Recycling
- Recycling EDRs in Pipeline-Only Systems

Registry Entries

Table 81-102 lists the FCT_PreCycle registry entries.

Table 81-102 FCT_PreCycle Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
RecycleSuffix	Specifies the suffix for the file that contains the EDRs that need recycling. The suffix is automatically appended when the file is moved from the reject directory to the input directory. If it is empty, no suffix is added. Default = **_Recy**	No
RecyFileName	Specifies the file name and path for the file that contains the EDRs that need recycling.	Yes

Sample Registry

```
PreRecycle
{
   ModuleName = FCT_PreRecycle
   Module
   {
      Active = True
      RecycleSuffix = RecycleFile
      RecyFileName = ./recycle.dat
```

```
    }
}
```

Semaphore File Entries

When you update the registry, you must select one of the following entries listed in Table 81-103:

Table 81-103 FCT_PreRecycle Semaphore File Entries

Entry	Description
Recycle	The module runs in real processing mode You can specify a list of files to recycle. If this entry is empty, all files from the reject directory are recycled.
RecycleTest	The module runs in test mode You can specify a list of files to test recycling with. If this entry is empty, all files from the reject directory are recycled.

Sample Semaphore File Entries

- Recycle all files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.Recycle {}
```

- Recycle only specific files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.Recycle.
File = ./format_a/abc.cdr

ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.Recycle.
File = ./format_a/xyz.cdr
```

- Test recycle all files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.RecycleTest {}
```

- Test recycle only specific files:

```
ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.RecycleTest.
File = ./format_a/abc.cdr

ifw.Pipelines.PRE_RECYCLE.Functions.Processing.FunctionPool.PreRecycle.Module.RecycleTest.
File = ./format_a/xyz.cdr
```

EDR Container Fields

Table 81-104 lists the FCT_PreRecycle EDR container fields.

Table 81-104 FCT_PreRecycle EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Write	Contains the internal process status.

Table 81-104 (Cont.) FCT_PreRecycle EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TRANSACTION_ID INTERNAL.TRANSACTION_ID	Decimal	Read	Contains the transaction ID.
STREAM_NAME INTERNAL.STREAM_NAME	String	Read	Contains the stream name.
PROCESS_STATUS INTERNAL.PROCESS_STATUS	Integer	Write	Contains the process status.

FCT_PreSuspense

This module is used both by the standard recycling mechanism and by the Suspense Manager service integration component that you purchase separately. Both implementations are described below.

> **Note:**
>
> This module stores the contents of the EDR before any other modules change it. This module must take the original version of an EDR as input, so that it can be recycled after being suspended.

Standard Recycling Implementation

The BRM FCT_PreSuspense module adds suspense-related information to EDRs. It adds the DETAIL.ASS_SUSPENSE_EXT data block to the EDR if that data block does not already exist.

Suspense Manager Implementation - Adding Queryable Fields

When used with Suspense Manager, FCT_PreSuspense configures the queryable fields for EDRs suspended in a specific pipeline. You must enter the table and field names from the ***/ suspended_usage*** object, as well as the corresponding EDR container fields. See "Registry Entries" for syntax and formatting information.

If no **QueryableFields** registry entry is present, the HEADER.QUERYABLE_FIELDS_MAPPING and DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS_MAPPING are set to empty strings.

> **Note:**
>
> Each table listed in the FCT_PreSuspense registry must also be configured in the RE Loader **Infranet.properties** file so that RE Loader can load into these tables.

This module adds queryable field mapping information to the HEADER.QUERYABLE_FIELDS_MAPPING field of the EDR.This information is passed to the Suspended Event (SE) Loader to generate control files for loading suspended usage records.

> **Note:**
>
> You add one set of queryable fields representing one **/suspended_usage** subclass *per pipeline*. For example, for a single pipeline that accepts **/suspended_usage/telco/gsm** records, you can pick queryable fields from the **/suspended_usage/telco** and **/suspended_usage/telco/gsm** subclasses. You could not pick queryable fields from **/suspended_usage/telco/gprs**, because it requires a separate pipeline.

FCT_PreSuspense serializes the original EDR container and stores it in DETAIL.ASS_SUSPENSE_EXT.EDR_BUF. It also stores the EDR size in DETAIL.ASS_SUSPENSE_EXT.EDR_SIZE.

When call records are being recycled, this module sets values for:

- HEADER.BATCH_ID, based on value already set by INP_Recycle in pre-recycle pipeline. This ID is appended with information to ensure that it remains unique.
- DETAIL.BATCH_ID with the batch ID value from the header record.

Changing the Way Batch IDs Are Set

You use the **KeepExistingBatchIds** registry entry to determine whether an EDR's batch ID is preserved as it is processed by the pipeline. A value of **False** directs the pipeline to change the batch ID; a value of **True** preserves it.

Set **KeepExistingBatchIds** to **False** (the default value) if the current pipeline is not part of a chain. Also set this entry to **False** if the current pipeline is the first pipeline in a chain of pipelines that includes the FCT_PreSuspense module.

Set this entry to **True** if the current pipeline is part of a chain of pipelines, and it is not the first pipeline in the chain that includes FCT_PreSuspense.

Dependencies

This module must be the first preprocessing module in a pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-105 lists the FCT_PreSuspense registry entries.

Table 81-105 FCT_PreSuspense Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes

Table 81-105 (Cont.) FCT_PreSuspense Registry Entries

Entry	Description	Mandatory
KeepExistingBatchIds	A value of **True** preserves the Batch ID in the detail record of each EDR in an EDR file. A value of **False** sets the Batch ID of each EDR to the Batch ID contained in the header record of the batch input file. The default is **False**.	No
QueryableFields (Suspense Manager only)	Specifies which fields in which tables are queryable from the Suspense Management Center. Includes the EDR container fields that correspond to the database fields. This entry is only useful to customers who have purchased Suspense Manager. Format: `QueryableFields` `{` `  table_name_1` `  {` `  database_column_name_1 = edr_container_field_1` `  database_column_name_2 = edr_container_field_2` `  }` `  table_name_2` `  {` `  database_column_name_3 = edr_container_field_3` `  database_column_name_4 = edr_container_field_4` `  }` `}` If this entry is not present, this module sets HEADER.QUERYABLE_FIELDS_MAPPING and DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS to empty strings.	Yes

Sample Registry

```
#---------------------------
# PreSuspense FCT
#---------------------------
PreSuspense
{
  ModuleName                = FCT_PreSuspense
  Module
  {
    Active                  = True
    QueryableFields
    {
      # table name. If more than one table, use a separate block
      SUSP_USAGE_TELCO_INFO_T
      {
        # format : <database_column_name> = <edr_conatiner_field_name>
        BYTES_IN = DETAIL.VOLUME_RECEIVED
        BYTES_OUT = DETAIL.VOLUME_SENT
        CALLED_TO = DETAIL.B_NUMBER
        #CALLING_FROM = DETAIL.B_NUMBER
        CALL_DURATION = DETAIL.DURATION
        PRIMARY_MSID = DETAIL.A_NUMBER
```

```
        SERVICE_TYPE = DETAIL.BASIC_SERVICE
        START_TIME = DETAIL.CHARGING_START_TIMESTAMP
        USAGE_TYPE = DETAIL.USAGE_TYPE
      }
      #Examples for GPRS calls
      #SUSP_USAGE_TELCO_GPRS_INFO_T
      #{
        # format : <database_column_name> = <edr_conatiner_field_name>
        #APN = DETAIL.ASS_GPRS_EXT.APN_ADDRESS
        #GGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.GGSN_ADDRESS
        #NODE_ID = DETAIL.ASS_GPRS_EXT.NODE_ID
        #SECONDARY_MSID = DETAIL.ASS_GPRS_EXT.PORT_NUMBER
        #SGSN_ADDRESS = DETAIL.ASS_GPRS_EXT.SGSN_ADDRESS
      #}
      #SUSP_USAGE_TELCO_GSM_INFO_T
      #{
        #APN = DETAIL.ASS_GSMW_EXT.APN_ADDRESS
        #CELL_ID = DETAIL.ASS_GSMW_EXT.CELL_ID
        #DESTINATION_SID = DETAIL.ASS_GSMW_EXT.TERMINATING_SWITCH_IDENTIFICATION
        #DIALED_NUMBER = DETAIL.ASS_GSMW_EXT.DIALED_DIGITS
        #ORIGIN_SID = DETAIL.ASS_GSMW_EXT.ORIGINATING_SWITCH_IDENTIFICATION
        #SECONDARY_MSID = DETAIL.ASS_GSMW_EXT.PORT_NUMBER
      #}
    }
  }
}
```

Semaphore File Entries

Table 81-106 lists the FCT_PreSuspense Semaphore file entry.

Table 81-106 FCT_PreSuspense Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.PreSuspense.Module.Active = False
```

EDR Container Fields

Table 81-107 lists the FCT_PreSuspense EDR container fields.

Table 81-107 FCT_PreSuspense EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
QUERYABLE_FIELDS_MAPPING HEADER.QUERYABLE_FIELDS_MAPPING	String	Write	Database column names and data types that map queryable fields. This field is used by Suspense Manager only. This is an empty string for standard recycling implementations.
PIPELINE_NAME DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME	String	Write	The name of the pipeline, derived from the pipeline registry.
SOURCE_FILENAME DETAIL.ASS_SUSPENSE_EXT.SOURCE_FILENAME	String	Write	The source file name. The same as INTERNAL.STREAM_NAME.
EDR_BUF DETAIL.ASS_SUSPENSE_EXT.EDR_BUF	String	Write	A stored representation of the EDR container with its original field values.
EDR_SIZE DETAIL.ASS_SUSPENSE_EXT.EDR_SIZE	Integer	Write	The size of DETAIL.ASS_SUSPENSE_EXT.EDR_BUF.
QUERYABLE_FIELDS DETAIL.ASS_SUSPENSE_EXT.QUERYABLE_FIELDS	String	Write	The queryable field values defined in the registry. This field is used by Suspense Manager only. This is an empty string for standard recycling implementations.
BATCH_ID DETAIL.BATCH_ID	String	Write	At recycling, the value is set from HEADER.BATCH_ID.
INTERN_PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Indicates whether the EDR is being recycled or test recycled.
ORIGINAL_BATCH_ID DETAIL.ORIGINAL_BATCH_ID	String	Write	Set the first time EDRs go through the pipeline (not being recycled or rerated.) Sets the value from HEADER.BATCH_ID in the header record.
BATCH_ID HEADER.BATCH_ID	String	Read	At recycling, modifies this value to ensure that it is unique.

FCT_RateAdjust

The FCT_RateAdjust module adjusts the charge for an EDR after rating has been performed.

Dependencies

If the rate adjustment data is stored in the database, the module requires a connection to the Pipeline Manager database.

This module must run after the FCT_MainRating module to adjust the rate.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-108 lists the FCT_RateAdjust registry entries.

Table 81-108 FCT_RateAdjust Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes, if the data is stored in the database. Otherwise not used.
RateAdjustFile	Specifies file name that contains the rate adjustment data. See "Creating a Rate Adjustment Rules File". You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise not used.
Source	Specifies where the rate adjustment data is stored: • **File** • **Database**	Yes

Sample Registry for the Database Interface

```
RateAdjust
{
  ModuleName = FCT_RateAdjust
  Module
  {
    Active = True
    Source = Database
    DataConnection = ifw.DataPool.DataConnection
  }
}
```

Sample Registry for the File Interface

```
RateAdjust
{
  ModuleName = FCT_RateAdjust
  Module
  {
    Active = True
    Source = File
    RateAdjustFile = /data/etc/discount.dat
```

```
    }
}
```

Semaphore File Entries

Table 81-109 lists the FCT_RateAdjust Semaphore file entries.

Table 81-109 FCT_RateAdjust Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
RateAdjustFile	Specifies file name that contains the rate adjustment data.
Reload	Reloads data from the database.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.RateAdjustment.Module.Active = False
```

EDR Container Fields

Table 81-110 lists the FCT_RateAdjust EDR container fields.

Table 81-110 FCT_RateAdjust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_CBD DETAIL.ASS_CBD	Block-Index	Read	Data block.
ASS_CBD_CHARGE_PACKET DETAIL.ASS_CBD.CP	Block-Index	Read	Data block.
ASS_CBD_RECORD_NUM DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Contains the record number.
USAGE_CLASS DETAIL.USAGE_CLASS	String	Read	Contains the usage class.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Read	Contains the usage type.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class.
BDR_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
DURATION DETAIL.DURATION	Decimal	Read	Contains the duration of the event.

Table 81-110 (Cont.) FCT_RateAdjust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network.
DESTINATION_NETWORK DETAIL.DESTINATION_NETWORK	String	Read	Contains the destination network.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the charge code.
INTERN_RATEPLAN_VERSION DETAIL.ASS_CBD.CP.INTERN_RATEPLAN_VERSION	Integer	Read	Contains the charge version.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the charge amount value.

Database Tables

The FCT_RateAdjust module uses the IFW_RATEADJUST table to set the rate adjustment rules.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Recycle

The FCT_Recycle module runs at the end of the pipeline It does either of the following:

- When the FCT_PreRecycle module runs in test mode, the FCT_Recycle module creates a report about the processing, but does not send the EDRs to an output file.
- When the FCT_PreRecycle module runs in recycle mode, the FCT_Recycle module sends the results to an output file, and attaches a sequence number to the output file.

See:

- Configuring Standard Recycling
- Recycling EDRs in Pipeline-Only Systems

Dependencies

> **Note:**
>
> You must configure the FCT_Recycle module as the last module of all function modules in the pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-111 lists the FCT_Recycle registry entries.

Table 81-111 FCT_Recycle Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
RecycleLog	Specifies the log file parameters: • **MessageFilePath** Specifies the path where the log file can find the message database. • **MessageFilePrefix** Specifies the prefix for collecting the files from the message file path. • **MessageFileSuffix** Specifies the suffix for collecting the files from the message file path. • **FilePath** Specifies the path in which the log file is written. • **FilePrefix** Specifies the prefix for the log file. • **FileSuffix** Specifies the suffix for the log file.	Yes

Sample Registry

```
Recycle
{
   ModuleName = FCT_Recycle
   Module
   {
      Active = True
      RecycleLog
      {
         MessageFilePath = ..
         MessageFilePrefix = Framework
         MessageFileSuffix = msg
         FilePath = ../tmp/log01
         FilePrefix = rej_
```

```
            FileSuffix = .log
        }
    }
}
```

Semaphore File Entries

Table 81-112 lists the FCT_Recycle Semaphore file entry.

Table 81-112 FCT_Recycle Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Recycle.Module.Active = True
```

EDR Container Fields

Table 81-113 lists the FCT_Recycle EDR container fields.

Table 81-113 FCT_Recycle EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
STREAM_NAME INTERNAL.STREAM_NAME	String	Read	Contains the stream name.
SEQ_CHECK INTERNAL.SEQ_CHECK	Integer	Write	Specifies the sequence check.
SEQ_GENERATION INTERNAL.SEQ_GENERATION	Integer	Write	Specifies the sequence generation.
OFFSET_GENERATION INTERNAL.OFFSET_GENERATION	Integer	Write	Specifies the offset generation.
PROCESS_STATUS INTERNAL.PROCESS_STATUS	Integer	Read	Contains the internal process status.

FCT_Reject

The FCT_Reject module analyzes the errors in an EDR and, if necessary, moves the EDR to a reject file.

See:

- Configuring Standard Recycling
- Recycling EDRs in Pipeline-Only Systems

Dependencies

This module must run after the rating and discount modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-114 lists the FCT_Reject registry entries.

Table 81-114 FCT_Reject Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
CallAssemblingModule	Provides data to the FCT_CallAssembling module to ensure that assembled calls are processed completely. See "Recycling Assembled EDRs".	No
MinErrorSeverity	Specifies to reject EDRs that have a specified severity. To allow warning and normal messages without rejecting the EDR, set this entry to 3. Default = Not used. See "Processing EDRs with Errors".	Yes
NotifyonReject	Specifies if other modules should be notified if an EDR is rejected. Default = **True**	No
StreamMap	Specifies a list of error types mapped to output streams. Important: A **UseRejectStream** entry is required to use **StreamMap**. See "Specifying Multiple Reject Streams".	No
UseRejectStream	Specifies whether to use the reject output stream: • **True**. Rejected EDRs are sent to the reject stream. • **False**. Rejected EDRs are sent to the normal output stream, but flagged as discarded.Important: A **StreamMap** entry is required to use **UseRejectStream**. See "Using a Reject Output Stream".	No

Sample Registry

```
Reject
{
   ModuleName = FCT_Reject
   {
         Active = True
         NotifyOnReject = True
         UseRejectStream = True
         CallAssemblingModule = ifw.Pipelines.Pipe.Functions.Standard.FunctionPool.CallAssembling
         StreamMap
               {
               error_type_1 = output_stream_1
               error_type_2 = output_stream_2
```

```
                intern = RejectStream
                logic = ReturnStream
                }
   }
}
```

Semaphore File Entries

Table 81-115 lists the FCT_Reject Semaphore file entry.

Table 81-115 FCT_Reject Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Rejection.Module.Active = False
```

Sample Output Configuration

You configure the reject stream in the registry in two places:

- In the pipeline configuration, for example:

```
Pipelines
{
ALL_RATE
{
Active   =   TRUE
.
.
.
RejectStream   =   RejectOutput
```

- In the Output configuration, for example:

```
# Output stream for rejected events
RejectOutput
{
 ModuleName = OUT_Reject
 Module
 {
  OutputStream
  {
   ModuleName = EXT_OutFileManager
   Module
   {
    OutputPath = ./samples/wireless/data/rej
    OutputPrefix = test
    OutputSuffix = .rej
    TempPrefix   = tmp

    TempDataPath    = ./samples/wireless/data/rej
    TempDataPrefix  = rej.tmp.
    TempDataSuffix  = .data
```

```
      Replace = TRUE
      DeleteEmptyFile = TRUE
    }
  }
 }
} # end of Reject
```

See "Configuring Output for Rejected or Duplicate EDRs".

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

EDR Container Fields

Table 81-116 lists the FCT_Reject container fields.

Table 81-116 FCT_Reject Container Fields

Field name	Access	Description
DISCARDING DETAIL.DISCARDING	Read/Write	**Read:** This field is used to detect pre-rejected EDRs. If this field is 1, the EDR is always rejected. **Write:** If this field is 0 and the EDR is rejected, the field is set to 1.
ERROR_REJECT_TYPE DETAIL.ERROR_REJECT_TYPE	Read	Specifies the type of error in the EDR. This determines which stream the EDR is directed to.

FCT_Rounding

The FCT_Rounding module performs rounding for rating and discounting. Add this module to the pipeline after the processing module for which it is rounding.

The FCT_Rounding pipeline module converts the rounding mode in the **/config/beid** object into the corresponding rounding mode used by the BAS rounding method in Pipeline Manager. If you use the BAS Decimal rounding method in custom pipeline modules and iScripts, you should include the FCT_Rounding module in your pipeline. Otherwise, you will need to convert the rounding mode in the **/config/beid** object into the BAS rounding mode before calling the BAS rounding method.

Dependencies

Requires a connection to the DAT_Currency module.

This module must run after the FCT_RateAdjust module if you want rating results to be rounded and after FCT_Discount module if you want discount results to be rounded. FCT_Rounding must come after each module for which rounding should occur. For batch rating, it must come before the FCT_ApplyBalance module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-117 lists the FCT_Rounding registry entries.

Table 81-117 FCT_Rounding Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
CurrencyDataModule	Specifies the connection to the DAT_Currency module.	Yes
Mode	Specifies the process for which rounding is applied: **Rating** = Round the balance impact of rating. **Taxation** = Round the balance impact of taxation. **Discounting** = Round the balance impact of discounting.	Yes

Sample Registry

```
Rounding
{
   ModuleName = FCT_Rounding
   Module
   {
      Active = TRUE
      Mode   = Rating
      CurrencyDataModule = ifw.DataPool.CurrencyDataModule
   }
}
```

EDR Container Fields

Table 81-118 lists the FCT_Rounding EDR container fields.

Table 81-118 FCT_Rounding EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
EVENT_TYPE DETAIL.EVENT_TYPE	String	Read	Specifies the event type.
ASS_CBD_RECORD_NUMBER DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read	Specifies the record number.
ASS_CBD_RECORD_TYPE DETAIL.ASS_CBD.RECORD_TYPE	String	Read	Specifies the record type.
RESOURCE_ID DETAIL.ASS_CBD.DP.RESOURCE_ID	Decimal	Read	Specifies the balance element ID.
CHARGED_AMOUNT_CURRENCY DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	Decimal	Read	Specifies the currency of the charged amount.

Table 81-118 (Cont.) FCT_Rounding EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CHARGED_AMOUNT_VALUE DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Integer	Write	Specifies the rounded charged amount.
TAX_VALUE DETAIL.ASS_CBD.TP.TAX_VALUE	Decimal	Write	Specifies the rounded tax value.
GRANTED_AMOUNT DETAIL.ASS_CBD.DP.GRANTED_AMOUNT	Integer	Write	Specifies the rounded noncurrency discount amount.
RELATED_CHARGE_INFO_ID DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Integer	Read	Specifies the index of the corresponding charge packet for which the tax was calculated.

FCT_RSC_Map

The FCT_RSC_Map module performs rate service class (RSC) mapping.

See "About Rate-Service Class Mapping".

Dependencies

Requires a connection to the Pipeline Manager database.

This module must run before FCT_MainRating module to change the Service Code and Service Class fields of the Charge Packet. These fields are used by the main rating module to find out the rate to be applied for a particular call.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-119 lists the FCT_RSC_Map registry entries.

Table 81-119 FCT_RSC_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
DefaultRSCGroup	Specifies the default RSC group to use when the RSC group is not specified in the EDR.	Yes

Sample Registry

```
RSC_Mapping
{
  ModuleName = FCT_RSC_Map
  Module
  {
    Active = True
    DataConnection = integrate.DataPool.DataConnection
    DefaultRscGroup = testGroup
  }
}
```

Semaphore File Entries

Table 81-120 lists the FCT_RSC_Map Semaphore file entries.

Table 81-120 FCT_RSC_Map Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.RateServiceClassMap.
Module.Active = False
```

EDR Container Fields

Table 81-121 lists the FCT_RSC_Map EDR container fields.

Table 81-121 FCT_RSC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging start time.
INTERN_SLA_RSC_GROUP DETAIL.INTERN_SLA_RSC_GROUP	String	Read	Contains the internal RSC group.
QOS_REQUESTED DETAIL.QOS_REQUESTED	String	Read	Contains the quality of service requested.
QOS_USED DETAIL.QOS_USED	String	Read	Contains the quality of service used.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Read	Contains the usage type.

Table 81-121 (Cont.) FCT_RSC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class.
INTERN_RATEPLAN DETAIL.ASS_CBD.CP.INTERN_RATEPLAN	String	Read	Contains the internal charge.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category.
SERVICE_CODE_USED DETAIL.ASS_CBD.CP.SERVICE_CODE_USED	String	Write	Contains the service code used.
SERVICE_CLASS_USED DETAIL.ASS_CBD.CP.SERVICE_CLASS_USED	String	Write	Contains the service class used.

Database Interface

FCT_RSC_Map uses the following database tables:

- **IFW_RSC_MAP**. Stores the mapping rules. See "About Rate-Service Class Mapping".
- The **IFW_RSC_GROUP**. Stores the RSC groups used for RSC mapping.
- The **IFW_SERVICECLASS**. Stores the service class codes used when defining service codes.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_SegZoneNoCust

The FCT_SegZoneNoCust module finds the segment using the source network information instead of using the customer information.

Dependencies

Requires a connection to the Pipeline Manager database.

This module must run before the FCT_MainZoning module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-122 lists the FCT_SegZoneNoCust registry entries.

Table 81-122 FCT_SegZoneNoCust Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
Segments	Specifies a list of mapping rules. Each rule defines the connection between the source network and the segment.	Yes

Sample Registry

```
SegZoneNoCust
{
  ModuleName = FCT_SegZoneNoCust
  Module
  {
    Active = True
    DataConnection = ifw.DataPool.Database
    Segments
    {
       26201 = SegmentD1
       26202 = SegmentD2
    }
  }
}
```

Semaphore File Entries

Table 81-123 lists the FCT_SegZoneNoCust Semaphore file entries.

Table 81-123 FCT_SegZoneNoCust Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SegZoneNoCust.Active = False
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SegZoneNoCust.Module.Reload { }
```

EDR Container Fields

Table 81-124 lists the FCT_SegZoneNoCust EDR container fields.

Table 81-124 FCT_SegZoneNoCust EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ASS_ZBD DETAIL.ASS_ZBD	Block	Create	Data block.
ASS_ZBD_ZONE_PACKET DETAIL.ASS_ZBD.ZP	Block	Create	Data block.
SOURCE_NETWORK DETAIL.SOURCE_NETWORK	String	Read	Contains the source network.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_ZBD_RECORD_TYPE DETAIL.ASS_ZBD.RECORD_TYPE	String	Write	Contains the record type.
ASS_ZBD_SEGMENT_CODE DETAIL.ASS_ZBD.SEGMENT_CODE	String	Write	Contains the segment code.
ASS_ZBD_SERVICE_CODE DETAIL.ASS_ZBD.SERVICE_CODE	String	Write	Contains the resulting service code.
ASS_ZBD_INTERN_ZONE_MODEL DETAIL.ASS_ZBD.ZP.INTERN_ZONE_MODEL	Integer	Write	Contains the zone model.

Database Tables

The FCT_SegZoneNoCust module uses the **IFW_SEGZONE_LNK** database table. This table stores segments used for multi-segment zoning.

FCT_ServiceCodeMap

The FCT_ServiceCodeMap module maps external service codes to internal service codes.

Dependencies

Requires a connection to the Pipeline Manager database.

Some modules require an internal service code, so this module should run near the front of a pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-125 lists the FCT_ServiceCodeMap registry entries.

Table 81-125 FCT_ServiceCodeMap Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
MapGroup	Specifies the map group that the service code map belongs to. You can use this entry in a semaphore file.	Yes

Sample Registry

```
ServiceCodeMapping
{
  ModuleName = FCT_ServiceCodeMap
  Module
  {
     Active = True
     DataConnection = integrate.DataPool.Database
     MapGroup = serviceMapGroup
  }
}
```

Semaphore File Entries

Table 81-126 lists the FCT_ServiceCodeMap Semaphore file entries.

Table 81-126 FCT_ServiceCodeMap Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
MapGroup	Specifies the mapping rule set.
Reload	Reloads data from the database into memory.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module.Reload {}
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.ServiceCodeMap.Module.MapGroup =
ALL_RATE
```

EDR Container Fields

The FCT_ServiceCodeMap module reads data from the EDR to map the external service code to the internal service code. The module then writes the internal service code and service class to the EDR.

Table 81-127 lists the FCT_ServiceCodeMap EDR container fields.

Table 81-127 FCT_ServiceCodeMap EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BASIC_SERVICE DETAIL.BASIC_SERVICE	String	Read	Contains the external service code.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Write	Contains the internal service code.
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Write	Contains the internal service class.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class.
ASS_GSMW_LOCATION_AREA_INDICATOR DETAIL.ASS_GSMW_EXT.LOCATION_AREA_INDICATOR	String	Read	Contains the GSMW extension location area.
ASS_GPRS_LOCATION_AREA_INDICATOR DETAIL.ASS_GPRS_EXT.LOCATION_AREA_INDICATOR	String	Read	Contains the GPRS extension location area.
QOS_REQUESTED DETAIL.QOS_REQUESTED	String	Read	Contains the quality of service requested.
QOS_USED DETAIL.QOS_USED	String	Read	Contains the quality of service used.
BDR_RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the record type.

Database Tables

The FCT_ServiceCodeMap module uses the following database tables:

- **IFW_SERVICE_MAP**. Maps external service codes to internal service codes.
- **IFW_MAP_GROUP**. Stores the map groups used for service code mapping.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_SocialNo

The FCT_SocialNo module flags social numbers for special processing. See "Setting Up Social Numbers".

Dependencies

If the social number data is stored in the database, the FCT_SocialNo module requires a connection to the Pipeline Manager database.

This module can run anywhere.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-128 lists the FCT_SocialNo registry entries.

Table 81-128 FCT_SocialNo Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	No
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes, if the data is stored in the database. Otherwise not used.
FileName	Specifies file that contains the social number data. See "Creating a Social Number Data File".	Yes, if the data is stored in a file.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. The default is **False**.	Yes
SocialNoMapSize	Specifies the size of the in-memory map that stores social numbers. For a large set of social numbers to be loaded, specifying this parameter will enhance loading performance.	No
Source	Specifies where the social number data is stored: • **File** • **Database**	Yes

Sample Registry for the Database Interface

```
SocialNo
{
  ModuleName = FCT_SocialNo
  Module
  {
    ReuseOnFailure = false
    Active = true
    Source = database
    DataConnection = dataPool
  }
}
```

Sample Registry for the File Interface

```
SocialNo
{
  ModuleName = FCT_SocialNo
  Module
  {
    Active = True
    ReuseOnFailure = False
    Source = File
    FileName = ../daten/socialno.dat
```

```
}
```

Semaphore File Entries

Table 81-129 lists the FCT_SocialNo Semaphore file entries.

Table 81-129 FCT_SocialNo Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
FileName	Reloads data from the specified file if **Source** parameter is set to **File**.
Reload	Reloads data from the database if **Source** parameter is set to **Database**.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.SocialNo.Module.Active = False
```

EDR Container Fields

The FCT_SocialNo module uses the following EDR container fields listed in Table 81-130 to flag social numbers for further processing.

Table 81-130 FCT_SocialNo EDR Container Fields.

Alias Field Name Default Field Name	Type	Access	Description
B_MODIFICATION_INDICATOR DETAIL.B_MODIFICATION_INDICATOR	String	Read/Write	Contains the modification indicator.
B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number.

Database Interface

The FCT_SocialNo module uses the **IFW_SOCIALNUMBER** database table. This table stores social numbers. See "Setting Up Social Numbers".

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Suspense

This module is used both by the standard recycling mechanism and by the Suspense Manager service integration component that you purchase separately. Both implementations are described below.

> **Note:**
>
> With one exception, FCT_Suspense must be the last function module in the pipeline. This ensures that it processes the final EDR container, including overwritten field values and enrichment field values. However, if FCT_AggreGate is used, it can be after FCT_Suspense.

Standard Recycling Implementation

As part of the standard recycling mechanism, the BRM FCT_Suspense function module:

- Routes EDRs being recycled from SuspenseCreateOutput to suspenseUpdateOutput.
- Logs the results of test recycling (if the **LogTestResults** registry entry is set).

Suspense reason and subreason codes are not supported with standard recycling, and these codes are all set to **O** (other).

Suspense Manager Implementation

As part of Suspense Manager, this module adds suspense reason and suspense subreason codes to EDRs. The specific errors that it adds are based on the error codes assigned to the EDR by the pipeline and the mapping information stored in the **/config/suspense_reason_code** object. If no **/config/suspense_reason_code** object is present, this module sets the suspense reason to **O** (other).

Dependencies

Requires a connection to the BRM database.

This module must be the last one in the pipeline.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-131 lists the FCT_Suspense registry entries.

Table 81-131 FCT_Suspense Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the BRM database.	Yes
LogTestResults	For standard recycling only. Determines whether the results of test recycling are logged. If this entry is not present, the results are not logged. If set to **True**, the **RecycleLog** entry must also be present in the FCT_Suspense registry. Default = **False** See "pin_recycle".	No
RecycleLog	Specifies the log file parameters.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.MessageFilePath	Specifies the path where the log file can find the message database.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.MessageFilePrefix	Specifies the prefix for collecting the files from the message file path.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.MessageFileSuffix	Specifies the suffix for collecting the files from the message file path.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.FilePath	Specifies the path in which the log file is written.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.FilePrefix	Specifies the prefix for the log file.	Yes, when **LogTestResults** is set to **True**.
RecycleLog.FileSuffix	Specifies the suffix for the log file.	Yes, when **LogTestResults** is set to **True**.
SuspenseCreateStream	Specifies the output stream for newly suspended EDRs.	Yes
SuspenseUpdateStream	Specifies the output stream for recycled EDRs.	Yes

Sample Registry

```
#---------------------------
# Suspense FCT
#---------------------------
Suspense
{
  ModuleName                = FCT_Suspense
  Module
  {
    Active                  = True
    SuspenseCreateStream    = SuspenseCreateOutput
    SuspenseUpdateStream    = SuspenseUpdateOutput
```

```
      EdrFieldMap              = DETAIL.ASS_GSMW_EXT.PORT_NUMBER
      DataConnection           = ifw.DataPool.LoginInfranet
      LogTestResults           = True
      RecycleLog
      {
         MessageFilePath = ..
         MessageFilePrefix = Framework
         MessageFileSuffix = msg
         FilePath = ../tmp/log01
         FilePrefix = rej_
         FileSuffix = .log
      }
    }
  }
```

Semaphore File Entries

Table 81-132 lists the FCT_Suspense file entry.

Table 81-132 FCT_Suspense Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.Suspense.Module.Active = False
```

EDR Container Fields

Table 81-133 lists the FCT_Suspense EDR container fields.

Table 81-133 FCT_Suspense EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ERROR_CODE DETAIL.ASS_SUSPENSE_EXT.ERROR_CODE	String	Write	The error code for the most severe error reported by the pipeline for the EDR.
SUSPENSE_REASON DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_REASON	String	Write	The suspense reason. Mapped from the error code. Used by Suspense Manager only. This field is set to **0** for standard recycling implementations.
SUSPENSE_SUBREASON DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_SUBREASON	String	Write	The suspense subreason. Mapped from the error code. Used by Suspense Manager only. This field is set to **0** for standard recycling implementations.

Table 81-133 (Cont.) FCT_Suspense EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
EDR_BUF DETAIL.ASS_SUSPENSE_EXT.EDR_BUF	String	Write	A stored representation of the EDR container including fields overwritten and enriched by the pipeline.
EDR_SIZE DETAIL.ASS_SUSPENSE_EXT.EDR_SIZE	Integer	Write	The size of DETAIL.ASS_SUSPENSE_EXT.EDR_BUF.
PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Read	Indicates whether the EDR is being recycled or test recycled.
SUSPENSE_STATUS DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_STATUS	Integer	Write	Indicates whether the EDR is suspended or successfully recycled.
BATCH_ID DETAIL.BATCH_ID	String	Write	Writes the batch ID from the header record, except during recycling.
BATCH_ID HEADER.BATCH_ID	String	Read	Written during recycling.

FCT_Timer

The FCT_Timer module sets the timer ID for an EDR and stores a copy of the EDR when the original EDR is sent to the processing pipeline.

If an EDR times out, FCT_Timer sets the timeout flag to **True** and sends:

- The original EDR with a timeout flag to the exception pipeline, which is the pipeline to which the original EDR is sent when it times out.
- The duplicate EDR to the timeout pipeline, which is the pipeline to which the duplicate EDR is sent when it times out.

If the EDR is processed within the time limit, FCT_Timer sets the timeout flag to **False**.

FCT_Timer also handles heartbeat and keep-alive messages by automatically resetting the timer at the interval specified in the **KeepAliveInterval** registry entry when there is message traffic between the CM and the Intelligent Network (IN).

Dependencies

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-134 lists the FCT_Timer registry entries.

Table 81-134 FCT_Timer Registry Entries

Entry	Description	Mandatory
Active	Specifies whether the module is active or inactive **True** = Active (default). **False** = Inactive.	Yes
Threads	Specifies the number of threads running this module. Default = **1**.	Yes
Reactors	Specifies the number of reactor objects to use. The reactors detect timeout events and then dispatch the events to the timer handler. The recommended number of reactors is from 1 to 5.	No
TimeoutQueue	Specifies the pipeline queue to which the duplicate EDR is sent when the EDR times out.	Yes
KeepAliveInterval	Specifies the idle timeout value in milliseconds, which specifies how long to wait before sending the original EDR to the exception pipeline. Default = **3000**.	Yes
Timeout	Specifies the idle timeout value in microseconds, which specifies how long to wait before sending the duplicate EDR to the timeout pipeline. Default = **100000**.	Yes
NoOpcodeNumbers	Specifies the number of the BRM opcode that prevents duplicate EDRs from being created. When the OPCODE_NUM EDR field is set to the specified number, the module does not create a duplicate EDR for the Timeout pipeline. Opcode numbers are defined in header (***.h**) files in the *BRM_home***/include/ops** directory. *BRM_home* is the directory where you installed BRM components.	No

Sample Registry

```
Timer
{
  ModuleName = FCT_Timer
   Module
  {
   Active = TRUE
   Threads = 1
   Reactors = 3
   TimeoutQueue = ifw.IPCQueues.TimeoutQueue
   KeepAliveInterval = 3000 ### milliseconds
   Timeout = 100000 ### microseconds
   NoOpcodeNumbers = 5000, 50001
   }
}
```

EDR Container Fields

FCT_Timer uses the EDR container fields listed in Table 81-135:

Table 81-135 FCT_Timer EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
TIMER_ID DETAIL.TIMER_ID	Integer	Write	ID assigned to the EDR container's timer when FCT_SetTimer schedules it. This ID is required to cancel the timer.
CURRENT_TIME DETAIL.CURRENT_TIME	Integer	Read	Current system time
TIMEOUT_OFFSET DETAIL.TIMEOUT_OFFSET	Integer	Write	Offset from the current system time
OPCODE_NUM DETAIL.OPCODE_NUM	Integer	Read	Specifies the number of the BRM opcode that performs the requested action. Opcode numbers are defined in header (***.h**) files in the *BRM_home***/include/ops** directory.
TIMEOUT_FLAG DETAIL.TIMEOUT_FLAG	Integer	Write	Specifies whether the EDR has timed out: • **0** is False. • **1** is True.
SESSION_ID DETAIL.SESSION_ID	String	Write	ID required to cancel the timer.
MILLISEC_TIME DETAIL.MILLISEC_TIME	Integer	Read	The latency time in milliseconds.

FCT_TriggerBill

The FCT_TriggerBill module sends EDRs to the billing-trigger output stream to trigger immediate billing for the associated accounts. It also sets a billing-trigger error code used to route the EDRs to the suspense output stream, and the **Trigger_Billing** recycle key used to retrieve the suspended EDRs for recycling.

Dependencies

Configure the FCT_TriggerBill module to run before the FCT_MainRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-136 lists the FCT_TriggerBill registry entries.

Table 81-136 FCT_TriggerBill Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
TriggerBillCreateStream	Specifies the billing-trigger output stream module.	Yes

Sample Registry

```
TriggerBill
{
   ModuleName = FCT_TriggerBill
   Module
   {
      Active = TRUE
      TriggerBillCreateStream = TriggerBillCreateOutput
   }
}
```

Semaphore File Entries

Table 81-137 lists the FCT_TriggerBill Semaphore file entry.

Table 81-137 FCT_TriggerBill Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

EDR Container Fields

Table 81-138 lists the FCT_TriggerBill EDR container fields.

Table 81-138 FCT_TriggerBill EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
ACTG_NEXT_DATE DETAIL.CUST.A.ACTG_NEXT_DATE	String	Read	The date that the current monthly cycle ends. Used to determine the accounting cycle to which the EDR belongs.
BILL_STATE DETAIL.CUST.A.BILL_STATE	String	Read	The billing state. Possible values are: • **PIN_ACTG_CYCLE_OPEN** (unbilled) • **PIN_ACTG_CYCLE_CLOSED** (billed)
CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	String	Read	The timestamp when the call started. Used to determine the accounting cycle to which the EDR belongs.
ERROR_CODE DETAIL.ASS_SUSPENSE_EXT.ERROR_CODE	String	Write	Specifies the billing-trigger error code. Used to send the EDR to the suspense output stream.
RECYCLE_KEY DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY	String	Write	The key value that identifies the EDRs suspended for pipeline-triggered billing. Set to **Trigger_Billing** when BILL_STATE is unbilled and ACTG_NEXT_DATE is passed.

Table 81-138 (Cont.) FCT_TriggerBill EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	X(5)	Read	The UTC time offset.

FCT_UoM_Map

The FCT_UoM_Map module converts the unit of measurement (UoM) of an incoming EDR to a UoM needed for rating a particular service.

Dependencies

Requires a connection to the Pipeline Manager database.

Must run after the FCT_ServiceCodeMap module, and before the rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-139 lists the FCT_UoM_Map registry entries.

Table 81-139 FCT_UoM_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
Mapping	Specifies the mapping rules.	Yes
Mapping.AssCBDServiceCode	Specifies the service code field in the associated charge breakdown records that is used for the mapping.	No
Mapping.InternServiceCode	Specifies the service code field in the basic detail block that is used for the mapping.	No

Sample Registry

```
UoM_Map
{
  ModuleName = FCT_UoM_Map
  Module
  {
    Active = True
    DataConnection = integrate.DataPool.Login
    Mapping
    {
      InternServiceCode = INTERN_SERVICE_CODE
```

```
        AssCBDServiceCode = ASS_CBD_SERVICE_CODE
      }
    }
  }
```

Semaphore File Entries

Table 81-140 lists the FCT_UoM_Map Semaphore file entries.

Table 81-140 FCT_UoM_Map Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
Reload	Reloads data from the database into memory.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UoM_Map.Module.Active = False
```

EDR Container Fields

Table 81-141 lists the FCT_UoM_Map EDR container fields.

Table 81-141 FCT_UoM_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
ASS_CBD_SERVICE_CODE DETAIL.ASS_CBD.SERVICE_CODE	String	Read	Contains the charge breakdown service code.
DETAIL.ASSOCIATED_CHARGE.CHARGE_PACKET ASS_CBD_CHARGE_PACKET	Struct	Read/Write	Contains charge packet data.

Database Interface

FCT_UoM_Map accesses the following database tables:

- **IFW_SERVICE**. This table stores data about services and associated RUMs.
- **IFW_RUMGROUP_LNK**. This table defines a list of RUM/UOM pairs with the RUM group value obtained from the **IFW_SERVICE** table.
- **IFW_UOM_MAP**. This table maps a UoM to a basic detail or to an associated charge packet.
- **IFW_UOM**. This table stores the UoMs for pipeline rating.
- **IFW_ALIAS_MAP**. This table stores an alias name for each RUM and UoM.

FCT_UsageClassMap

The FCT_UsageClassMap module maps external codes for supplementary services, such as call forwarding, to internal usage classes.

Dependencies

Requires a connection to the Pipeline Manager database.

The FCT_UsageClassMap module is run before the zoning and rating modules.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-142 lists the FCT_UsageClassMap registry entries.

Table 81-142 FCT_UsageClassMap Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
MapGroup	Specifies the map group. You can use this entry in a semaphore file.	Yes
OverwriteUsageClass	Specifies if the external usage class should be overwritten by the internal one. The default is to not overwrite the external usage class; if you map usage codes, you should enable this entry. Default = **False**	No
OptimizeFor	Specifies if the module is configured to optimize memory consumption as well as pipeline startup time. **Memory** = Optimizes memory consumption and pipeline startup time. **No memory optimization** = Does not optimize memory consumption and pipeline startup time (the default). **Note** • Enabling this entry might have an adverse impact on the number of call detail records (CDRs) processed in a specific time interval. • This entry is read only at pipeline start up. Its value cannot be changed by using a semaphore.	No

Sample Registry

```
UsageClassMapping
{
  ModuleName = FCT_UsageClassMap
  Module
  {
    Active  = True
    DataConnection  =  ifw.DataPool.Database
```

```
    OverwriteUsageClass = False
    MapGroup  = mapGroup0
    OptimizeFor = Memory
  }
}
```

Semaphore File Entries

Table 81-143 lists the FCT_UsageClassMap Semaphore file entries.

Table 81-143 FCT_UsageClassMap Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
MapGroup	Specifies the mapping rule set.
Reload	Reloads data from the database into memory.

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module.Reload {}
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageClassMap.Module.MapGroup =
ALL_RATE
```

EDR Container Fields

The FCT_UsageClassMap module adds the internal usage class to the EDR. All other fields in Table 81-144 are used for mapping the usage class.

Table 81-144 FCT_UsageClassMap EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_RECORD_TYPE DETAIL.RECORD_TYPE	String	Read	Contains the event record type.
USAGE_CLASS DETAIL.USAGE_CLASS	String	Read	Contains the external usage class.
USAGE_TYPE DETAIL.USAGE_CLASS	String	Read	Contains the external usage type.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read	Contains the wholesale impact category.
TARIFF_CLASS DETAIL.TARIFF_CLASS	String	Read	Contains the tariff class.
TARIFF_SUB_CLASS DETAIL.TARIFF_SUB_CLASS	String	Read	Contains the tariff subclass.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Write	Contains the internal usage class.

Table 81-144 (Cont.) FCT_UsageClassMap EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
CONNECT_TYPE DETAIL.CONNECT_TYPE	String	Read	Contains the connection type.
CONNECT_SUB_TYPE DETAIL.CONNECT_SUB_TYPE	String	Read	Contains the connection subtype.
APN_ADDRESS DETAIL.ASS_GPRS_EXT.APN_ADDRESS	String	Read	Contains the GPRS APN type.
ACTION_CODE DETAIL.ASS_GSMW_EXT.SS_PACKET.ACTION_CODE	String	Read	Contains the GSM SS packet action code.
SS_EVENT DETAIL.ASS_GSMW_EXT.SS_PACKET.SS_EVENT	String	Read	Contains the GSM SS packet action event.
INTERN_C_NUMBER_ZONE DETAIL.INTERN_C_NUMBER_ZONE	String	Read	Contains the internal normalized C number.
DETAIL.ASS_GPRS_EXT ASS_GPRS	String	Read	Contains the GPRS extension.
DETAIL.ASS_GSMW_EXT.SS_PACKET ASS_GSMW_SS_PACKET	String	Read	Contains the GSMW SS packet.

Database Tables

The FCT_UsageClassMap module uses the following database tables:

- The **IFW_USAGECLASS_MAP** table maps external supplementary service codes in the EDR to internal usage classes.
- The **IFW_USAGECLASS** table stores the usage classes that can be used as a result of usage class mapping.
- The **IFW_MAP_GROUP** table stores the map groups used for usage class mapping.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_USC_Map

The FCT_USC_Map module performs usage scenario mapping.

Dependencies

This module needs a connection to the DAT_USC_Map module.

This module must run after the following:

- FCT_UsageClassMap
- ISC_UsageType
- FCT_PreRating

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-145 lists the FCT_USC_Map registry entries.

Table 81-145 FCT_USC_Map Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_USC_Map data module.	Yes
DefaultUSCGroup	Specifies the USC group that contains the mapping rules. If no matching rule is found, the FCT_USC_Map module uses the rule in the default USC map group. You can use this entry in a semaphore file.	Yes
LogZoneModelNotFoundEntries	Specifies, if set to **True**, that all log entries in INF_NO_USC_MAPPING_ENTRY are logged into the Stream log. The default value is **False**.	No
Mode	Specifies the mode in which USC mapping is done. The **Rating** mode (the default) specifies that USC mapping is done using the zone model from the charge packets. Mapping in this mode provides the impact category for charge packets. The **Zoning** mode specifies that USC mapping is done using the zone model from the EDR detail block. Mapping in this mode provides impact categories for the detail block. Using the **Zoning** mode requires that the DETAIL.RETAIL_IMPACT_CATEGORY and DETAIL.WHOLESALE_IMPACT_CATEGORY fields are populated.	No

Sample Registry

```
USC_Mapping
{
  ModuleName = FCT_USC_Map
  Module
  {
    Active = True
    DefaultUSCGroup = usc_group
    DataModule = ifw.DataPool.USCDataModule
  }
}
```

Semaphore File Entries

Table 81-146 lists the FCT_USC_Map Semaphore file entry.

Table 81-146 FCT_USC_Map Semaphore File Entry

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive

Sample Semaphore File Entry

```
ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageScenarioMap.
Module.Active = True

ifw.Pipelines.ALL_RATE.Functions.Processing.FunctionPool.UsageScenarioMap.
Module. DefaultUSCGroup  =  usc_group
```

EDR Container Fields

Table 81-147 lists the FCT_USC_Map EDR container fields.

Table 81-147 FCT_USC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	String	Read	Contains the charging time stamp.
INTERN_SLA_USC_GROUP DETAIL.INTERN_SLA_USC_GROUP	String	Read	Contains the internal USC group.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Read/Write	Contains the usage type code. This field is updated only when the **Mode** registry entry is set to **Zoning**. It is not updated when the **Mode** entry is set to **Rating**.
RETAIL_IMPACT_CATEGORY DETAIL.RETAIL_IMPACT_CATEGORY	String	Read/Write	Contains the retail impact category.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read/Write	Contains the wholesale impact category.
WHOLESALE_CHARGED_AMOUNT_VALUE DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	String	Read	Contains the wholesale charged amount value.
IC_DESCRIPTION DETAIL.IC_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
IC_DESCRIPTION DETAIL.ASS_CBD.CP.IC_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.

Table 81-147 (Cont.) FCT_USC_Map EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
INTERN_SERVICE_CLASS DETAIL.INTERN_SERVICE_CLASS	String	Read	Contains the internal service class.
DURATION DETAIL.DURATION	String	Read	Contains the duration of the event.
INTERN_USAGE_CLASS DETAIL.INTERN_USAGE_CLASS	String	Read	Contains the internal usage class code.
BDR_INTERN_ZONE_MODEL DETAIL.INTERN_ZONE_MODEL	String	Read	Contains the zone model.
RATEPLAN_TYPE DETAIL.ASS_CBD.CP.RATEPLAN_TYPE	String	Read	Contains the charge breakdown record charge type.
ASS_CBD_INTERN_ZONE_MODEL DETAIL.ASS_CBD.CP.INTERN_ZONE_MODEL	String	Read	Contains the charge breakdown record zone model.
ASS_CBD_IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read/Write	Contains the charge breakdown record impact category.
ASS_CBD_ZONE_ENTRY_NAME DETAIL.ASS_CBD.CP.ZONE_ENTRY_NAME	String	Read	Contains the zone name used for the charge packet.
ZONE_ENTRY_NAME DETAIL.ZONE_ENTRY_NAME	String	Read	Contains the zone name of the event.

Database Interface

The FCT_USC_Map module uses the following database tables:

- IFW_USC_MAP. This table stores mapping rules for usage scenario maps.
- IFW_USC_GROUP. This table stores USC group codes used for usage scenario mapping.
- IFW_USAGETYPE. This table stores usage type codes used for usage scenario mapping.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

FCT_Zone

The FCT_Zone module computes zones when you use Pipeline Manager only for zoning.

Dependencies

The FCT_Zone module requires a connection to the DAT_Zone module.

This module must run after FCT_Account.

For more information, see "Function Module Dependencies".

Registry Entries

Table 81-148 lists the FCT_Zone registry entries.

Table 81-148 FCT_Zone Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive You can use this entry in a semaphore file.	Yes
DataModule	Specifies the connection to the DAT_Zone module.	Yes
EdrZoneModel	Specifies the zone model which should be used for the zoning. You can use this entry in a semaphore file.	Yes

Sample Registry

```
Zoning
{
  ModuleName = FCT_Zone
  Module
  {
    Active = True
    DataModule = ifw.DataPool.ZoneDataModule
    EdrZoneModel = ZM_ADD
  }
}
```

Semaphore File Entries

Table 81-149 lists the FCT_Zone Semaphore file entries.

Table 81-149 FCT_Zone Semaphore File Entries

Entry	Description
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive
EdrZoneModel	Specifies the zone model which should be used for the zoning.

Sample Semaphore File Entry

```
ifw.Pipelines.Functions.Processing.FunctionPool.Zoning.Module.EdrZoneModel = D2_FUN
```

EDR Container Fields

Table 81-150 lists the FCT_Zone EDR container fields.

Table 81-150 FCT_Zone EDR Container Fields

Alias Field Name Default Field Name	Type	Access	Description
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the charging time stamp.
RETAIL_IMPACT_CATEGORY DETAIL.RETAIL_IMPACT_CATEGORY	String	Write	Contains the retail impact category.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Write	Contains the wholesale impact category.
ZONE_DESCRIPTION DETAIL.ZONE_DESCRIPTION	String	Write	Contains the zone description for displaying on invoices.
ZONE_ENTRY_NAME DETAIL.ZONE_ENTRY_NAME	String	Write	Contains the destination description for displaying on invoices.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Contains the internal service code.
INTERN_A_NUMBER_ZONE DETAIL.INTERN_A_NUMBER_ZONE	String	Read	Contains the A number.
INTERN_B_NUMBER_ZONE DETAIL.INTERN_B_NUMBER_ZONE	String	Read	Contains the B number.
BDR_INTERN_ZONE_MODEL DETAIL.INTERN_ZONE_MODEL	Integer	Write	Contains the resulting zone model ID.
BDR_INTERN_APN_GROUP DETAIL.INTERN_APN_GROUP	String	Write	Contains the zone model related APN_GROUP for use by the FCT_APN_Map module.

82
Pipeline Manager Data Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager data modules.

DAT_AccountBatch

DAT_AccountBatch retrieves account data from the BRM database for the "DAT_ItemAssign", "FCT_Account", and "FCT_AccountRouter" modules.

This module also maintains a list of the accounts that are being rerated by the **pin_rerate** utility. This information is used by the batch rating pipeline to suspend incoming call detail records (CDRs) for those accounts while rerating is in progress.

Dependencies

This module requires connections to the following:

- BRM database.
- Pipeline Manager database.
- DAT_Listener module. See "DAT_Listener".
- DAT_PortalConfig module. See "DAT_PortalConfig".

Registry Entries

Table 82-1 lists the DAT_AccountBatch registry entries.

Table 82-1 DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
AcceptLoginSearchFailure	If set to **True,** when a customer login number is not found in memory, the EDR will be accepted, the pipeline will continue processing the EDR, and a warning will be reported in the stream log. If set to **False** (the default), when a customer login number is not found in memory, the EDR will be set as invalid and rejected and a major error will be reported. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
AccountLocks	Use this entry to tune performance by managing thread contention. Default = **10**	No
AddAliasList	Specifies whether all alias names and logins are added to the EDR. Default = **False** **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No

Table 82-1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
ClosedAccountDelay	Specifies to not load closed accounts. Also specifies the number of days prior to the current date for which closed accounts are not loaded. For example, if **ClosedAccountDelay** is set to **10** and the current date it June 20, accounts that were closed prior to June 10 are not loaded into memory. Default = **0**	No
Connections	Specifies the number of connections to the database. This value must be at least the number of threads plus 1. Default = **5**	No
EnrichRatingProductOnly	If set to **True**, only the purchased charge offers whose service ID matches the service ID in the EDR being rated are added to the EDR. If set to **False**, the purchased charge offers whose service type matches the service type in the EDR being rated are added to the EDR. **Note:** This entry is present for backward compatibility.	No
InfranetConnection	Specifies the connection to the BRM database.	Yes
InitialLoading	Specifies whether the initial loading of service and account data is performed. Otherwise, loading occurs while processing. Login objects are always loaded. Setting this entry to **False** enables the system to start faster. Default = **True** **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
IntegrateConnection	Specifies the connection to the Pipeline Manager database.	Yes
Listener	Specifies the connection to the DAT_Listener module.	Yes
LoadAccountForSharingOnly	Specifies whether Pipeline Manager can load serviceless accounts that are owners of resource sharing groups. Default = **False**	No
LoadLogins	Specifies whether the login is loaded in case of an existing alias list. When set to **True**, logins are loaded from both the PIN_FLD_LOGIN field and the PIN_FLD_ALIAS_LIST array. When set to **False**, logins are only loaded from the PIN_FLD_ALIAS_LIST array. Default = **False** when **UseAsRouter** is disabled. When **UseAsRouter** is enabled, then **LoadLogins** is always **True**.	No
LoadPercentage	Indicates the percentage of account POIDs to store locally when determining the account blocks for which each thread is responsible. Values must be greater than 0.000000 and less than or equal to 100.0. Default = **10.0**	Yes
LogEvents	Specifies whether received events should be written to a log file. Use this entry to troubleshoot Pipeline Manager event handling. Default = **False**	No

Table 82-1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
LoginLocks	Use this entry to tune performance by managing thread contention. Default = **10**	No
PerThreadJobsCount	Specifies the number of jobs per thread. **Important:** Setting the number of jobs per thread to a large number can decrease performance because of the system overhead associated with creating too many jobs. (Typically, three to eight jobs per thread is optimal). If you want to adjust the number of accounts or balances per job, increase or decrease the number of threads.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables DAT_AccountBatch to retrieve business parameter settings from the DAT_PortalConfig module.	Yes
ProfilesNotLoadedIntoEdr	When you have accounts that own or share a large number of ERA profiles, use this registry entry to specify the profiles to be filtered. When a CDR is rated, if the value in the EDR field matches a value in one or more of the specified ERA profiles, the ISC_ProfileLabel iScript loads only the ERA labels into the EDR container. See "Improving Pipeline Rating Performance for Events with ERAs".	No
ReadAccountBalances	Specifies whether to load account resource data. The data includes resource IDs, such as 840. If enabled, the RESOURCE_LIST field in the CUSTOMER_DATA block is populated with the resource IDs for that account. Default = **False** **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ReadAllProducts	If set to **True,** all the purchased charge offers for the account are added to the EDR. If set to **False**, only those purchased charge offers matching the service types and event types of the CDR processed are added to the EDR.	No
ReadPlans	If set to **True**, the module loads plan IDs into memory when loading purchased charge offers. During EDR processing, the list of plan IDs for an account is returned to the FCT_Account module.	No
ReadSystemProductFromMain	If set to **True,** the module retrieves the latest system charge offers from the main tables and uses the start and end dates to validate when the charge offer is in effect. If set to **False** (the default), the module retrieves the system charge offer information from the audit tables and uses the purchase creation date to validate when the charge offer is in effect. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
RejectClosedAccounts	Specifies whether to reject CDRs for accounts that are closed. If set to **True**, all closed account information is loaded from the database. Any CDR with a timestamp later than the account's closed date is rejected. Default = **False**	No

Table 82-1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
RowFetchSize	Specifies the number of rows of data to retrieve from the BRM database. Use this entry for performance tuning. Default = **1000**	No
ServiceLocks	Tunes performance by managing thread contention. Default = **10**	No
ThreadAccountHashMapSize	Controls the size of the temporary hash map built by each thread for accounts. **Important:** • Changing the default system-calculated values for this entry is not recommended. • If the **UseAsRouter** registry entry is set to **True,** this registry entry is not used.	No
ThreadGroupSharingChargesHashMapSize	Controls the size of the temporary hash map built by each thread for loading charge share group data. The system-calculated default value might not be appropriate. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ThreadGroupSharingDiscountsHashMapSize	Controls the size of the temporary hash map built by each thread for loading discount sharing group data. The system-calculated default value might not be appropriate. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ThreadGroupSharingMonitorsHashMapSize	Controls the size of a temporary hash map constructed for GroupSharingProfile object storage during multi-thread DAT_Account initialization. Default = (TotalAccounts / NumThreads) * 0.10. The default value for this entry is appropriate in most cases. However, the value should be increased if you exceed an average of 4 GroupSharingMonitors for every 10 accounts.	No
ThreadGroupSharingProfilesHashMapSizes	Controls the size of the temporary hash map built by each thread for loading profile sharing group data. The system-calculated default value might not be appropriate. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
ThreadLoginHashMapSize	Controls the size of the temporary hash map built by each thread for loading logins. The system-calculated default value is appropriate for most BRM implementations.	No
Threads	Specifies the number of threads. Set this value to at least the number of CPUs in the system. Increasing the number of threads increases performance, up to a point. Specifying too many threads decreases performance. Default = **4**	Yes
ThreadServiceHashMapSize	Controls the size of the temporary hash map built by each thread for loading services. The system-calculated default value is appropriate for most BRM implementations. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
TimesTenEnabled	Set this to **False**.	No

Table 82-1 (Cont.) DAT_AccountBatch Registry Entries

Entry	Description	Mandatory
UseAsRouter	If set to **True,** the module is used by the FCT_AccountRouter module to route EDRs to separate Pipeline Manager instances. See "Using Pipeline Manager with Multiple Database Schemas" and "FCT_AccountRouter". If set to **False** (the default), the module is used by the FCT_Account module. **Important:** If set to **True**, the following registry entries are not used: • AcceptLoginSearchFailure • AddAliasList • InitialLoading • ReadAccountBalances • ReadSystemProductFromMain • ThreadAccountHashMapSize • ThreadGroupSharingChargesHashMapSize • ThreadGroupSharingDiscountsHashMapSize • ThreadGroupSharingProfilesHashMapSizes • ThreadServiceHashMapSize • UseProductCreatedTime • UseLatestProductAndDiscount	No
UseLatestProductAndDiscount	If set to **True**, the module retrieves the latest purchased charge offer and discount offer information from the main tables and uses the start and end dates to validate when the charge offer is in effect. If set to **False** (the default), the module retrieves the purchased charge offer and discount offer information from the audit tables and uses the purchase creation date to validate when the charge offer is in effect. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
UseProductCreatedTime	If set to **True** (the default), the charge offer is selected only if an event occurs after the charge offer's created time (PIN_FLD_CREATED_T) and between its start and end times. If set to **False**, charge offer validity is checked based only on the start and end times (PIN_FLD_START_T and PIN_FLD_END_T) of the charge offer. **Important:** If the **UseAsRouter** registry entry is set to **True**, this registry entry is not used.	No
UseProfileEffectiveTime	If set to **True** (the default), the module uses EFFECTIVE_T to determine the validity of the profile objects. If set to **False**, the module uses CREATED_T to determine the validity of the profile objects.	No

Sample Registry Entry

```
CustomerData
  {
    ModuleName = DAT_AccountBatch
    Module
    {
      IntegrateConnection  = ifw.DataPool.Login
```

```
    InfranetConnection   = ifw.DataPool.LoginInfranet
    LogEvents            = True
    Listener             = ifw.DataPool.Listener
    TimesTenEnabled      = False
    ReadAccountBalances  = True
    Threads              = 4
    Connections          =  5
    LoadPercentage       = 10.0
  }
}
```

Semaphore File Entries

Table 82-2 lists the DAT_AccountBatch Semaphore file entries.

Table 82-2 DAT_AccountBatch Semaphore File Entries

Entry	Description
LogEvents	Specifies whether events should be stored in a log file. You can also use this entry in the startup registry.
PrintData	Reports the account data for all accounts.
PrintDataLogin	Reports the account data for a single account identified by the BRM login ID (usually the phone number).
PrintDataSamples	Reports the account data for a specified number of accounts, chosen randomly.
PrintAmtData	Prints in-memory data about the Account Migration Manager (AMM) to the specified log file.
PrintAmtJobData	Prints in-memory data about one account migration job to the specified log file.
RejectClosedAccounts	Rejects CDRs with a timestamp later than the account's closed date.
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.CustomerData.Module.Reload {}
```

Database Tables

The DAT_AccountBatch module uses the following database tables:

- IFW_CURRENCY
- IFW_REF_MAP
- IFW_SERVICE

DAT_AccountRealtime

The DAT_AccountRealtime module provides customer data from the BRM database in a real-time discounting pipeline.

> **Note:**
>
> Unlike the DAT_AccountBatch module, the DAT_AccountRealtime module does not load account data in memory when you start Pipeline Manager. Instead, it gets account data in real time from the BRM database by using the NET_EM module.

The DAT_AccountRealtime module gets data for the FCT_Discount module. For information about the FCT_Discount module, see "FCT_Discount".

Dependencies

The DAT_AccountRealtime requires the NET_EM module. It makes a connection to the NET_EM module automatically; you don't need to configure the connection.

Registry Entries

There are no registry entries for the DAT_AccountRealtime module. You only need to enter the module in the registry DataPool section.

Sample Registry Entry

```
CustomerData
  {
    ModuleName = Dat_AccountRealtime
    Module
    {
      #
    }
  }
```

Semaphore File Entries

DAT_AccountRealtime does not support semaphore updates.

DAT_BalanceBatch

The DAT_BalanceBatch module maintains balance information in the Pipeline Manager memory. It uses Account Synchronization to retrieve balance information from the BRM database. Data is stored in memory only, not in the database or in a file.

When reading balances and sub-balances from the database, the DAT_BalanceBatch module ignores balance monitor impacts.

See the following documents:

- Configuring Discounting Modules and Components
- FCT_Discount

Dependencies

Requires the following connections:

- Pipeline Manager database.
- BRM database.
- DAT_AccountBatch module. See "DAT_AccountBatch".
- DAT_Listener module. See "DAT_Listener".
- DAT_Discount module. See "DAT_Discount".
- DAT_PortalConfig module. See "DAT_PortalConfig".

Registry Entries

Table 82-3 lists the DAT_BalanceBatch registry entries.

Table 82-3 DAT_BalanceBatch Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountBatch module.	Yes
BalanceDirectory	Specifies the directory that contains data and transaction files.	No
BalanceLocks	Specifies the number of locks that can be acquired during processing. Must be a positive integer. Default = **100** **Important**: Setting this value too low may decrease pipeline throughput performance.	No
BalanceLockStatusLog	Specifies that when an event transaction is locked by an EDR transaction, it is logged to the process logger. Default = **False**	No
BalancesPerThreadJobsCount	Specifies the number of jobs per thread. **Important:** Setting the number of jobs per thread to a large number can decrease performance because of the system overhead associated with creating too many jobs. (Typically, three to eight jobs per thread is optimal). If you want to adjust the number of accounts or balances per job, you can do this by increasing or decreasing the number of threads.	Yes
BalanceTrace	Specifies whether to generate a balance trace file. **True** indicates that a balance trace file is generated. **False** indicates that a balance trace file is not generated. Default = **False**	No
CustomEvents	Lists custom business events that include balance data needed by Pipeline Manager. Custom events are defined in the Account Synchronization DM payload configuration file (**payloadconfig_ifw_sync.xml**). See "Configuring Custom Business Events for Pipeline Discounting".	No
DiscountDataModule	Specifies the connection to the DAT_Discount module.	Yes
InfranetConnection	Specifies the database connection to the BRM database.	Yes
IntegrateConnection	Specifies the database connection to the Pipeline Manager database.	Yes
ListenerDataModule	Specifies the connection to the DAT_Listener module.	Yes

Table 82-3 (Cont.) DAT_BalanceBatch Registry Entries

Entry	Description	Mandatory
LoadPercentage	Specifies how much data to load from the BRM database before the process log outputs status information. For example, to output status after every 10% of the data is loaded, enter **10**. Default = **10**	No
LogEvents	Specifies whether received events should be written to a log file. Use this entry to troubleshoot Pipeline Manager event handling. Default = **False**	No
LogTransactions	Specifies if the balances affected during the CDR processing are logged. Default = **False**	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables DAT_BalanceBatch to retrieve business parameter settings from the DAT_PortalConfig module.	Yes
RowFetchSize	Specifies the number of rows of balance data to load from the BRM database for each database retrieving. Default = **50**	No
SelectiveSubBalLoad	Specifies whether to selectively load noncurrency sub-balances at Pipeline Manager startup. Default = **True**	No
Synchronized	Specifies whether to allow the first transaction to process and to make other transactions wait in the queue. Default = **False**	No
ThreadHashMapSize	Specifies the size of the hash map in each thread used for loading balance data from the BRM database. Default = **1024**	No
Threads	Specifies the number of threads for loading the balance data from the BRM database. The number of threads must be smaller than or equal to the number of connections. Default = **4**	No
UseFlexibleConsumptionRule	Specifies whether to use the resource consumption rules defined at the package level. **True**: Uses the consumption rules defined for each resource in a balance group. If a consumption rule is not defined, this module uses the rules defined in the **/config/beid** object. If a consumption rule isn't defined in a balance group or the **/config/beid** object, this module uses the rule defined in the **multi_bal** instance of the **/config/business_params** object. **False**: Uses the system-wide consumption rule defined in the **multi_bal** instance of the **/config/business_params** object only. Default = **True**	No

Table 82-3 (Cont.) DAT_BalanceBatch Registry Entries

Entry	Description	Mandatory
VirtualTime	Specifies whether this module uses system time or virtual time. Default = **False** Set to **True** *only* if you are performing tests and have used the **pin_virtual_time** utility to set a virtual time. If you set this entry to **True**, make sure you copy the **pin.conf** file from the *BRM_home*/**sys**/**test** directory to the *pipeline_home* directory. *BRM_home* is the directory where you installed BRM components. The **pin.conf** file contains this entry: **- - pin_virtual_time pin_virtual_time_file**	No

Sample Registry Entry

```
BalanceDataModule
  {
    ModuleName = DAT_BalanceBatch
    Module
    {
      IntegrateConnection = ifw.DataPool.Login
      InfranetConnection  = ifw.DataPool.LoginInfranet
      AccountDataModule   = ifw.DataPool.CustomerData
      ListenerDataModule  = ifw.DataPool.Listener
      DiscountDataModule  = ifw.DataPool.DiscountData
      BalanceDirectory    = ./samples/wireless/data/balance
      UseFlexibleConsumptionRule = True
      CustomEvents
      {
        CycleRollover20days
      }
    }
  }
```

Semaphore File Entries

Table 82-4 lists the DAT_BalanceBatch Semaphore file entries.

Table 82-4 DAT_BalanceBatch Semaphore File Entries

Entry	Description
BalanceGroupId	Specifies the ID field of the balance group POID entry. The balance data referenced by **BalanceGroupId** is written into the file specified by the **DataFileName** entry.
DataFileName	Specifies the file name that contains balance data. If the **BalanceGroupId** entry is not present, DAT_BalanceBatch writes all balance data in memory into the file.
LogEvents	Specifies whether events should be stored in a log file. You can also use this entry in the startup registry.
ReloadCreditThresholdParam	Reloads the value from the **CreditThresholdChecking** business parameter.

Sample Semaphore File Entry

```
ifw.DataPool.BalanceDataModule.Module.DataFileName = balData.txt
ifw.DataPool.BalanceDataModule.Module.BalanceGroupId = 70015
ifw.DataPool.BalanceDataModule.Module.LogEvents = True
ifw.DataPool.BalanceDataModule.Module.ReloadCreditThresholdParam{}
```

DAT_BalanceRealtime

The DAT_BalanceRealtime module runs in a real-time discounting pipeline. It retrieves the current balance from the BRM database and supplies the data for real-time discounting.

> **Note:**
>
> Unlike the DAT_BalanceBatch module, the DAT_BalanceRealtime module does not load balance data in memory when you start Pipeline Manager. Instead, it gets balance data in real time from the BRM database by using the NET_EM module.

See the following documents:

- Configuring a Real-Time Discounting Pipeline
- FCT_Discount

Dependencies

The DAT_BalanceRealtime module requires the NET_EM module. It makes a connection to the NET_EM module automatically; you don't need to configure the connection.

Registry Entries

There are no registry entries for the DAT_BalanceRealtime module. You only need to enter the module in the registry DataPool section.

Sample Registry Entry

```
BalanceDataModule
  {
    ModuleName = DAT_BalanceRealtime
    Module
    {
      #
    }
  }
```

Semaphore File Entries

DAT_BalanceRealtime does not support semaphore updates.

DAT_Calendar

The DAT_Calendar module provides special day calendar data for the FCT_MainRating module.

Dependencies

Requires a connection to the Database Connect (DBC) module. See "Database Connect (DBC)".

Registry Entries

Table 82-5 lists the DAT_Calendar registry entries.

Table 82-5 DAT_Calendar Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database.	Yes

Sample Registry Entry

```
Calendar
{
  ModuleName = DAT_Calendar
  Module
  {
    DataConnection = ifw.DataPool.Login
  }
}
```

Semaphore File Entries

Table 82-6 lists the DAT_Calendar Semaphore file entries.

Table 82-6 DAT_Calendar Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.CalendarDataModule.Module.Reload {}
```

Events

Table 82-7 lists the DAT_Calendar events.

Table 82-7 DAT_Calendar Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_Calendar	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_Calendar	None

Database Tables

The DAT_Calendar module uses the following database tables:

- IFW_CALENDAR
- IFW_HOLIDAY

DAT_ConnectionMonitor

This module creates and monitors the idle timeout period for each connection and maintains the state for each client.

Registry Entries

Table 82-8 lists the DAT_ConnectionMonitor registry entries.

Table 82-8 DAT_ConnectionMonitor Registry Entries

Entry	Description	Mandatory
KeepAliveInterval	The idle timeout value in milliseconds, which specifies how long to wait for a message from the client before sending a Device Watchdog Request (DWR) message to the client. Default is **30000**.	Yes
KeepAliveQueue	Specifies the pipeline queue to which the dummy EDR for the DWR message is sent.	Yes
ShutdownInterval	The idle timeout value in milliseconds, which specifies how long to wait before shutting down after sending a Disconnect-Peer-Request (DPR) message to the client. Default is **1000**.	No
Threads	Number of threads in the pool. Default is **1**.	Yes

Sample Registry Entry

```
ConnectionMonitor
{
  ModuleName = DAT_ConnectionMonitor
  Module
  {
    Threads = 1
    KeepAliveInterval = 30000
    ShutdownInterval = 1000
    KeepAliveQueue = ifw.IPCQueues.INOutputQueue
```

```
    }
}
```

Semaphore File Entries

DAT_ConnectionMonitor does not support semaphore updates.

DAT_ConnectionPool

DAT_ConnectionPool module has a set of configured Connection Manager (CM) connections, which the "FCT_Opcode" module uses to connect to the CM and call the appropriate opcode.

For each CM, the DAT_ConnectionPool module maintains a queue for spare connections, determined by the size of the queue.

The DAT_ConnectionPool module balances the load by distributing the required load among the pipelines using the FCT_Opcode module and when there is a CM failure, redistributes the load among the active CMs.

> **Note:**
>
> If a connection fails, the processing pipeline connects to the spare connection of the CM to which it initially connected; if the CM is down, it tries to use a connection from a different CM queue.

If an idle processing pipeline is connected to an inactive CM, The DAT_ConnectionPool module sets the connection status to inactive and updates the pipeline recycle flag, so that when the pipeline starts processing a request it can connect to an active CM.

The DAT_ConnectionPool module initializes the Global Data Dictionary (GDD) during startup by accessing the database.

> **Note:**
>
> If CMs are not available, it uses a file containing the data dictionary flist to initialize the GDD.

Registry Entries

Table 82-9 lists the DAT_ConnectionPool registry entries.

Table 82-9 DAT_ConnectionPool Registry Entries

Entry	Description	Mandatory
InfranetDataDictionaryFileName	The file containing the data dictionary. If a CM is not available, DAT_ConnectionPool uses this file to start and initialize the GDD. If you don't specify a file name, the DAT_ConnectionPool uses the default file, **./gddDataFile.dat**, where it stored the data dictionary flists at the initial startup.	No

Table 82-9 (Cont.) DAT_ConnectionPool Registry Entries

Entry	Description	Mandatory
IdleConnectionBuffer	Size of the queue for spare connections.	Yes
FullQueueTimeout	The interval, in seconds, in which the worker thread pings the queue to check if there is space available in the queue for a connection, when the queue is full. Default is 10 seconds.	No
EmptyQueueTimeout	The interval, in seconds, in which the pipeline thread pings the queue to check if there is a connection available in the queue, when the queue is empty. It can happen during startup of the pipeline or when the CM connection is not working as expected, for example, the CM times out. Default is 1 second.	No
InfranetPool	Specifies the CMs in the connection pool. For each CM in the pool, define the following entries: • Host name (**Host =** *CM1_host)* • Port number (**Port =***CM1_port)* • Login name and password for logging into BRM (**LoginName =** *root.0.0.0.1* and • **LoginPassword =***password)* • Whether to log debug messages (**Logging =** *False*. Values are **True** and **False**. The default is **False**.) • CM response timeout in milliseconds (**SocketTimeOut =** *30000*.)	Yes

Sample Registry Entry

```
DataPool
{

   CMConnectionPool
  {
    ModuleName = DAT_ConnectionPool
    Module
    {
      InfranetDataDictionaryFileName = File_with_DD_objects
      IdleConnectionBuffer
      {
        Size = 2
       }
      FullQueueTimeout = number_of_seconds
      EmptyQueueTimeout = number_of_seconds
      InfranetPool
     {
       CM1
         {
           Host = CM1_host
           Port = CM1_port
           LoginName = root.0.0.0.1
           LoginPassword = password
           Logging = True
           SocketTimeOut = 30000
         }
      CM2
         {
```

```
            Host = CM2_host
            Port = CM2_port
            LoginName = root.0.0.0.1
            LoginPassword = password
            Logging = True
            SocketTimeOut = 30000
           }
      }
   }
}
```

Semaphore File Entries

DAT_ConnectionPool does not support semaphore updates.

DAT_Currency

The DAT_Currency module converts currency symbols to numeric values and retreives resource rounding rules, using data from **/config/beid** objects in the BRM database.

Dependencies

Requires a connection to the BRM database.

Registry Entries

Table 82-10 lists the DAT_Currency registry entries.

Table 82-10 DAT_Currency Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database.	Yes
InfranetConnection	Specifies the database connection to the BRM database.	Yes
ReuseOnFailure	Specifies whether the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	No

Sample Registry Entry

```
DAT_Currency
{
  ModuleName = DAT_Currency
  Module
  {
    ReuseOnFailure = TRUE
    InfranetConnection  = ifw.DataPool.LoginInfranet
    DataConnection = ifw.DataPool.Login
  }
}
```

Semaphore File Entries

Table 82-11 lists the DAT_Currency Semaphore file entries.

Table 82-11 DAT_Currency Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.CurrencyDataModule.Module.Reload ()
```

DAT_Dayrate

The DAT_Dayrate module provides special day rate data for the FCT_Dayrate module.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-12 lists the DAT_Dayrate registry entries.

Table 82-12 DAT_Dayrate Registry Entries

Entry	Description	Mandatory
Buffer	Specifies the size of the internal data buffer.	Yes
DataConnection	Specifies the database connection to the Pipeline Manager database.	Yes

Sample Registry Entry

```
Dayrate
{
  ModuleName = DAT_Dayrate
  Module
  {
    DataConnection = ifw.DataPool.Login
    Buffer = 5000
  }
}
```

Semaphore File Entries

Table 82-13 lists the DAT_Dayrate Semaphore file entries.

Table 82-13 DAT_Dayrate Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.DayRateDataModule.Module.Reload {}
```

Events

Table 82-14 lists the DAT_Dayrate events.

Table 82-14 DAT_Dayrate Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_Dayrate	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_Dayrate	None

Database Tables

The DAT_Dayrate module uses the following database tables:

- IFW_SPECIALDAYRATE
- IFW_SPECIALDAY_LNK

DAT_Discount

The DAT_Discount module provides discount data for the FCT_Discount module. See "FCT_Discount".

Dependencies

Requires a connection to:

- The Pipeline Manager database.
- The BRM database.
- The DAT_AccountRealtime or DAT_AccountBatch module.
- The DAT_ModelSelector module.
- The DAT_PortalConfig module.

Registry Entries

Table 82-15 lists the DAT_Discount registry entries.

Table 82-15 DAT_Discount Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountRealtime or DAT_AccountBatch module.	Yes
InfranetConnection	Specifies the database connection to the BRM database.	Yes
EvalScriptFiles	Specify name-value pairs for one or more iScript files. The name is a unique string used to identify the script if there is an error. The value is the relative or absolute path of the file. The iScript files specified in this entry contain functions that can be referenced via **EVAL** tokens in discount expressions. Any number of files can be specified.	No
IntegrateConnection	Specifies the database connection to the Pipeline Manager database.	Yes
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module. This enables DAT_Discount to retrieve business parameter settings from the DAT_PortalConfig module.	Yes

Sample Registry Entry

```
ifw
{ ....
  DataPool
  { ....
    DiscountModelDataModule
    {
      ModuleName = DAT_Discount
      Module
      {
        IntegrateConnection = ifw.DataPool.Login
        InfranetConnection = ifw.DataPool.LoginInfranet
        AccountDataModule = ifw.DataPool.CustomerData
        #Customizable iScript files supporting EVAL function
        EvalScriptFiles
        {
           scriptFile1 = ../iScriptLib/iScriptLib_Samples/ISC_GetMostCalledInfo.isc
           scriptFile2 = ../iScriptLib/iScriptLib_Samples/ISC_GetLastSixMonthCharge.isc
        }
      }
    }
  }
}
```

Semaphore File Entries

Table 82-16 lists the DAT_Discount Semaphore file entries.

Table 82-16 DAT_Discount Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database. **Note:** Discounting data is reloaded only when the discount configuration contains a new noncurrency resource.

Table 82-16 (Cont.) DAT_Discount Semaphore File Entries

Entry	Description
ReloadEvalScripts	Reloads and recompiles the iScript files specified in the **EvalScriptFiles** registry entry.
DiscountModel	The discounts whose data you want written to the output. This entry is used for troubleshooting purposes and must be used in conjunction with the **DataFileName** semaphore. • **ALL**: Writes all discount codes (such as model codes, rule codes, step codes, and so on) and related configuration information for all discounts in your system. • *Discount_model_code*: Writes the discount codes and related configuration information associated with the specified discount code. **Note:** You cannot specify multiple discount codes. You can specify only a single code or ALL.
DataFileName	Where discount information should be written. This entry is used for troubleshooting purposes and must be used in conjunction with the **DiscountModel** semaphore. • To write the information to a file, specify a file name. By default, the file is created in the *pipeline_home* directory. • To write the information to the terminal, leave the value of this entry blank.

Sample Semaphore File Entry

- To reload data from the database and to reload and recompile iScript files:

```
ifw.DataPool.DiscountModelDataModule.Module.Reload {}
ifw.DataPool.DiscountModelDataModule.Module.ReloadEvalScripts=True
```

- To write all discount configuration information to a file named **DiscountConfig.log**:

```
ifw.DataPool.DiscountModelDataModule.Module.DiscountCode = ALL
ifw.DataPool.DiscountModelDataModule.Module.DataFileName = DiscountConfig.log
```

- To write the configuration information for a discount with the code **DM10%off** to the terminal:

```
ifw.DataPool.DiscountModelDataModule.Module.DiscountCode = DM10%OFF
ifw.DataPool.DiscountModelDataModule.Module.DataFileName {}
```

Database Tables

The DAT_Discount module uses the following database tables:

- IFW_DISCOUNTMODEL
- IFW_DSCMDL_VER
- IFW_DSCMDL_CNF
- IFW_DSCTRIGGER
- IFW_DSCCONDITION
- IFW_DISCOUNTMASTER
- IFW_DISCOUNTDETAIL
- IFW_DISCOUNTRULE
- IFW_DISCOUNTSTEP

- IFW_DSCBALIMPACT

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_ExchangeRate

The DAT_ExchangeRate module provides currency exchange rate data for the FCT_ExchangeRate module.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-17 lists the DAT_ExchangeRate registry entries.

Table 82-17 DAT_ExchangeRate Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database.	Yes
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	No

Sample Registry Entry

```
ExchangeRateData
{
    ModuleName = DAT_ExchangeRate
    Module
    {
      DataConnection = ifw.DataPool.Login
      ReuseOnFailure = True
    }
}
```

Semaphore File Entries

Table 82-18 lists the DAT_ExchangeRate Semaphore file entries.

Table 82-18 DAT_ExchangeRate Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.ExchangeRateDataModule.Module.Reload {}
```

Events

Table 82-19 lists the DAT_ExchangeRate events.

Table 82-19 DAT_ExchangeRate Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_ExchangeRate	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_ExchangeRate	None

Database Tables

The DAT_ExchangeRate module uses the IFW_EXCHANGE_RATE database table.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_InterConnect

The DAT_InterConnect module caches InterConnect and roaming related configuration data. This information is used by FCT_CarrierIcRating module.

> **Note:**
>
> Use the OpFlagExt iScript extension in iScripts to call this module directly to obtain the network operator taxation value flag.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-20 lists the DAT_InterConnect registry entries.

Table 82-20 DAT_InterConnect Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database.	Yes
LoadPoiAreas	If **True**, load data from the IFW_POIAREA_LNK table for reseller interconnection network models.	No
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	Yes

Sample Registry Entry

```
InterConnect
{
  ModuleName = DAT_InterConnect
  Module
  {
    DataConnection = ifw.DataPool.Login
    ReuseOnFailure = False
  }
}
```

Semaphore File Entries

Table 82-21 lists the DAT_InterConnect Semaphore file entries.

Table 82-21 DAT_InterConnect Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.
LoadPoiAreas	Specifies if the module should load data from the IFW_POIAREA_LNK table for reseller interconnection network models.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.

Sample Semaphore File Entry

```
ifw.DataPool.InterConnect.Module.Reload {}
ifw.DataPool.InterConnectDataModule.Module.ReuseOnFailure = True
ifw.DataPool.InterConnectDataModule.Module.LoadPoiAreas = True
```

Database Tables

The DAT_InterConnect module uses the following database tables:

- IFW_NETWORKOPER
- IFW_NETWORKMODEL
- IFW_ICPRODUCT
- IFW_ICPRODUCT_RATE

- IFW_ICPRODUCT_GRP
- IFW_ICPRODUCT_CNF

Data for interconnect rating is stored in the following tables:

- IFW_SWITCH
- IFW_POI
- IFW_TRUNK
- IFW_TRUNK_CNF
- IFW_POIAREA_LNK

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_ItemAssign

The DAT_ItemAssign module returns the item POID for an item tag to the FCT_ItemAssign and FCT_Billing Record modules.

Dependencies

Requires a connection to the BRM database and the "DAT_AccountBatch" module.

Registry Entries

Table 82-22 lists the DAT_ItemAssign registry entries.

Table 82-22 DAT_ItemAssign Registry Entries

Entry	Description	Mandatory
AccountDataModule	Specifies the connection to the DAT_AccountBatch module.	Yes
InfranetConnection	Specifies the connection to the BRM database.	Yes
ItemPoidReservedRangeUnitSize	Specifies the maximum number of POIDs to be reserved. Default =**10000**	No

Sample Registry Entry

```
Flexible item assignment data module

ItemAssignDataModule
{
  ModuleName = DAT_ItemAssign
  Module
  {
   InfranetConnection = ifw.DataPool.LoginInfranet
   AccountDataModule = ifw.DataPool.CustomerData
```

```
    ItemPoidReservedUnitSize = 10000
  }
}
```

Semaphore File Entries

Table 82-23 lists the DAT_ItemAssign Semaphore file entries.

Table 82-23 DAT_ItemAssign Semaphore File Entries

Entry	Description
PrintData	Generates a log file that contains the item tag-to-type mapping information.
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.ItemAssignDataModule.Module.PrintData=TagTypeMap.txt
ifw.dataPool.ItemAssignDataModule.Module.Reload {}
```

DAT_Listener

The DAT_Listener module dequeues business events from a database queue and then sends the data to the DAT_AccountBatch and DAT_Discount modules.

The DAT_Listener module also posts acknowledgment events to the acknowledgment queue in response to business events sent by **pin_rerate**.

DAT_Listener module controls whether the Pipeline Manager processes business events or CDRs by interleaving the two processes. You can configure the DAT_Listener module for concurrent or interleaved processing.

Dependencies

Requires a connection to the database containing the account synchronization queue.

The Listener section of the registry file must be listed after the Pipeline Manager and BRM database connection sections. Otherwise, the Pipeline Manager fails to start.

Registry Entries

Table 82-24 lists the DAT_Listener registry entries.

Table 82-24 DAT_Listener Registry Entries

Entry	Description	Mandatory
AckQueueNameAMM	Specifies the name of the acknowledgment queue for posting AMM-related acknowledgment events.	Yes This entry is mandatory for AMM operations.

Table 82-24 (Cont.) DAT_Listener Registry Entries

Entry	Description	Mandatory
AckQueueName	Specifies the name of the acknowledgment queue for posting rerating related acknowledgment events. Default = **RERATING_ACK_QUEUE**	Yes This entry is mandatory for rerating using **pin_rerate**.
BatchSize	The maximum number of events to be dequeued in each dequeuing operation. Default = 10	No
EventsPath	Specifies the directory location of the file that stores the event information retrieved by DAT_Listener if the LogEvents entry is set to **True**. Default location is the directory where the ifw is launched.	No
EventsPrefix	Specifies the prefix of the name of the file that stores the event information retrieved by DAT_Listener if the LogEvents entry is set to **True**. Default =**listenerLog**	No
EventThreadAllocation	Defines the number of threads (in addition to one default thread) to use for dequeuing specific business events. For example: `EventThreadAllocation` `{` `  RecycleRequest = 1` `  OpenNewActgCycle = 2` `}` uses 4 threads: one for RecycleRequest business events, two for OpenNewActgCycle business events, and one default thread for dequeuing all other types of business events.	No
InfranetConnection	Specifies the connection to a database with the account synchronization queue. Default = **ifw.DataPool.LoginInfranet**	Yes
LogEvents	Specifies whether received events should be written to a log file. Default = **False**	No
NumOfRetries	Specifies the number of times the DAT_Listener module retries to connect to the database queue. Default = 10	No
QueueLibrary	Specifies whether the DAT_Listener module is configured for Oracle AQ. Set QueueLibrary to **OracleQueue**.	Yes
QueueName	Specifies the name of the database queue from which the DAT_Listener module retrieves events. Default = **IFW_SYNC_QUEUE**	Yes
RetryInterval	Specifies the time in seconds that the DAT_Listener module waits before trying to reconnect to the database specified in **InfranetConnection**. Default = **5**	No

Registry Entries for Interleaved Processing

The following are registry entries used for interleaved processing.

> **Note:**
>
> The default values for interleaved processing are also the minimum required values. If you specify a value less than the default for any entry, that value is ignored and the minimum default value is used.

Table 82-25 lists the DAT_Listener registry entries for interleaved processing.

Table 82-25 DAT_Listener Registry Entries for Interleaved Processing

Entry	Description	Mandatory
CheckInterval	Specifies (in seconds) how frequently the DAT_Listener module checks the number of events waiting in the queue. If this entry is not present, the default frequency check is used. **Important:** This entry takes precedence over **MaxNumEvents**, **MinNumEvents**, **MaxEventProcessTime**, and **MaxCDRProcessTime**. For example, if **MaxEventProcessTime** is set to 3600 seconds, and **CheckInterval** is set to 7200 seconds, events are processed for 7200 seconds. Default = **60**	No
EnableInterLeaving Statistics	Specifies whether to log only interleaving statistical data. If set to **False**, all processing messages are logged. Default = **False**	No
InterleavingReqd	Specifies whether interleaved processing is enabled: **True** = Enabled **False** = Not enabled When set to **False** or not specified, interleaved processing is not performed; CDRs and events are processed simultaneously. Default = **False**	No
MaxCDRProcessTime	Specifies the maximum number of seconds that CDRs are processed. When the pipeline has been processing CDRs for this amount of time, the DAT_Listener module stops CDR processing and starts business event processing regardless of how many business events are in the queue. Default and minimum allowed = **300**	If **MaxEventProcessTime** is specified and **InterleavingReqd** is set to TRUE, yes. Otherwise, no.
MaxEventProcessTime	Specifies the maximum number of seconds that business events are processed. When the pipeline has been processing business events for this amount of time, the DAT_Listener module stops business event processing and starts CDR processing regardless of how many business events are in the queue. Default and minimum allowed = **60**	If **MaxCDRProcessTime** is specified and **InterleavingReqd** is set to TRUE, yes. Otherwise, no.

Table 82-25 (Cont.) DAT_Listener Registry Entries for Interleaved Processing

Entry	Description	Mandatory
MaxNumEvents	Specifies the maximum number of business events allowed in the queue. When the number of events in the queue reaches or exceeds this amount, DAT_Listener stops pipeline CDR processing and starts business event processing. Default and minimum allowed = **900**	Yes, if **InterleavingReqd** is set to **True**. Otherwise, no. Requires that you also specify **MinNumEvents** and **MaxNumEvents** is greater than **MinNumEvents**.
MinNumEvents	Specifies the minimum number of business events allowed in the queue. When the number of events in the queue reaches or drops below this amount, the DAT_Listener stops business event processing and starts CDR processing. Default and minimum allowed = **300**	Yes, if **InterleavingReqd** is set to **True**. Otherwise, no.
ProcessAllEvents	Specifies whether to process all business events in the queue when Pipeline Manager is started: **True** = Processes all business events in the queue before activating interleaved processing. **False** = Interleaved processing is activated at startup. Business events are processed according to the interleaving settings. If set to **True** at startup, after processing all business events, this entry is reset to **False**. To process all business events at startup, you must reset this entry to **True** each time you restart Pipeline Manager. Default = **False**	No

Sample Registry Entry

```
Listener
{
    ModuleName = DAT_Listener
    Module
    {
       InfranetConnection = ifw.DataPool.LoginInfranet
       QueueLibrary = OracleQueue
       QueueName = IFW_SYNC_QUEUE
       NumOfRetries = 1
       RetryInterval = 5
       LogEvents = TRUE

       InterleavingReqd = true
       MaxNumEvents = 900
       MinNumEvents = 300
       CheckInterval = 60
       EnableInterLeavingStatistics = false
       ProcessAllEvents = true
       MaxEventProcessTime = 60
       MaxCDRProcessTime = 300
    }
}
```

Semaphore File Entries

Table 82-26 lists the DAT_Listener Semaphore file entries.

Table 82-26 DAT_Listener Semaphore File Entries

Entry	Description
AckQueueNameAMM	Specifies the name of the acknowledgment queue for posting AMM-related acknowledgment events. To add or modify this entry without having to stop the pipeline, use the **Disconnect** semaphore to disconnect the Listener before setting it. After specifying the queue name, use the **Connect** semaphore to reconnect the Listener to the pipeline for the new value to take effect.
CheckInterval	Specifies (in seconds) how frequently the DAT_Listener module checks the number of events waiting in the queue. If this entry is not present, the default frequency check is used.
Connect{}	Reconnects the DAT_Listener module event dequeuing threads to the Account Synchronization queue.
Disconnect{}	Disconnects the DAT_Listener module event dequeuing threads from the Account Synchronization queue. The module checks the dequeuing threads before disconnecting them: • If a thread is in the middle of processing an event, the module waits until the pipeline finishes processing events, and then suspends and disconnects the thread from the queuing database. • If a thread *is not* in the middle of processing an event, the module suspends and disconnects the thread from the queuing database.
EnableDequeueStatistics	Specifies whether to log dequeue statistics in the process log. **TRUE** = Log dequeue statistics **FALSE** = Do not log dequeue statistics (Default) **Note:** When set to **TRUE**, the size of the process log increases. Additionally, there may be a performance impact due to file input and output processing. Use this entry for diagnostic purposes only and should not be used otherwise.
EnableInterLeavingStatistics	Specifies whether to log only interleaving statistical data. If set to **False**, all processing messages are logged.
LogEvents	Specifies whether received events should be written to a log file. Default = **False**
MaxCDRProcessTime	Specifies the maximum number of seconds that CDRs are processed. When the pipeline has been processing CDRs for this amount of time, the DAT_Listener module stops CDR processing and starts business event processing regardless of how many business events are in the queue. Required when **MaxEventProcessTime** is specified. Requires that you also specify **MaxNumEvents**, **MinNumEvents**, and **MaxEventProcessTime**.
MaxEventProcessTime	Specifies the maximum number of seconds that business events are processed. When the pipeline has been processing business events for this amount of time, the DAT_Listener module stops business event processing and starts CDR processing regardless of how many business events are in the queue. Required when **MaxCDRProcessTime** is specified. Requires that you also specify **MaxNumEvents**, **MinNumEvents**, and **MaxCDRProcessTime**.

Table 82-26 (Cont.) DAT_Listener Semaphore File Entries

Entry	Description
MaxNumEvents	Specifies the maximum number of business events allowed in the queue. When the number of events in the queue reaches this amount, the DAT_Listener module stops pipeline CDR processing and starts business event processing. Required when **MinNumEvents**, **MaxEventProcessTime**, or **MaxCDRProcessTime** is specified. Requires that you also specify **MinNumEvents**.
MinNumEvents	Specifies the minimum number of business events allowed in the queue. When the number of events in the queue reaches this amount, the DAT_Listener module stops business event processing and starts CDR processing. Required when **MaxNumEvents**, **MaxEventProcessTime**, or **MaxCDRProcessTime** is specified. Requires that you also specify **MaxNumEvents**.

Sample Semaphore File Entry

```
ifw.DataPool.Listener.Module.CheckInterval=180
ifw.DataPool.Listener.Module.EnableInterLeavingStatistics=true
ifw.DataPool.Listener.Module.Disconnect{}
ifw.DataPool.Listener.Module.Connect{}
```

DAT_ModelSelector

When a model selector is used to rate or discount an EDR, the DAT_ModelSelector module evaluates the model selector rules to determine the correct price or discount. The rules are evaluated in the order they are ranked in the model selector.

The following rating and discounting modules get the model information from DAT_ModelSelector:

- FCT_MainRating gets the pricing from DAT_ModelSelector. See "FCT_MainRating".
- FCT_DiscountAnalysis gets the discount from DAT_ModelSelector. See "FCT_DiscountAnalysis".

Dependencies

Requires a connection to the Database Connect (DBC) module. See "Database Connect (DBC)".

The module uses event notification to refresh customized charge offer data. You must configure a connection to DAT_Listener if you plan to use this feature. See "DAT_Listener".

Registry Entries

Table 82-27 lists the DAT_ModelSelector registry entries.

Table 82-27 DAT_ModelSelector Registry Entries

Entry	Description	Mandatory
IntegrateConnection	Specifies the connection to the Pipeline Manager database. This typically points to the login registry section. For example: `IntegrateConnection = ifw.DataPool.Login`	Yes
ListenerDataModule	Specifies the connection to the DAT_Listener module.	No

Sample Registry Entry

```
ModelSelectorDataModule
    {
      ModuleName = DAT_ModelSelector
      Module
      {
     ListenerDataModule  = ifw.DataPool.Listener
        IntegrateConnection = ifw.DataPool.Login
        LogEvents           = True
      }
```

Semaphore File Entries

Table 82-28 lists the DAT_ModelSelector Semaphore file entries.

Table 82-28 DAT_ModelSelector Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.
DiscountModel	The discounts whose data you want written to the output. This entry is used for troubleshooting purposes and must be used in conjunction with the **DataFileName** semaphore. • **ALL**: Writes all discount codes (such as model codes, rule codes, step codes, and so on) and related configuration information for all discounts in your system. • *Discount_model_code*: Writes the discount codes and related configuration information associated with the specified discount code. **Note:** You cannot specify multiple discount codes. You can specify only a single code or **ALL**.
DataFileName	Where discount information should be written. This entry is used for troubleshooting purposes and must be used in conjunction with the **DiscountModel** semaphore. • To write the information to a file, specify a file name. By default, the file is created in the *pipeline_home* directory. • To write the information to the terminal, leave the value of this entry blank.

Sample Semaphore File Entry

To reload data from the database and to reload and recompile iScript files:

```
ifw.DataPool.ModelSelectorDataModule.Module.Reload {}
```

To write all model selector configuration information to a file named **ModelSelectorConfig.log**:

```
ifw.DataPool.ModelSelectorDataModule.Module.ModelSelectorCode = ALL

ifw.DataPool.ModelSelectorDataModule.Module.DataFileName = ModelSelectorConfig.log
```

To write the configuration information for a model selector with the code **DMS10%off** to the terminal:

```
ifw.DataPool.ModelSelectorDataModule.Module.ModelSelectorCode = DMS10%off
ifw.DataPool.ModelSelectorDataModule.Module.DataFileName {}
```

Database Tables

The DAT_ModelSelector module uses the following database tables:

- IFW_MODEL_SELECTOR. This table stores all model selector information in the Pipeline Manager database. It has a type field to indicate whether a model selector is for discounting or rating.
- IFW_SELECTOR_RULESET. This table maps model selector rules to specific model selectors. Rules associated with a model selector are ranked in order of priority.
- IFW_SELECTOR_RULE. This table stores information for each model selector rule, including the code, name, and rule links.
- IFW_SELECTOR_RULE_LNK. This table maps a model selector rule to its detail or block.
- IFW_SELECTOR_DETAIL. This table stores each model selector's rule details; the EDR field and value.
- IFW_SELECTOR_BLOCK. This table stores block information for a model selector rule. This table is for future use.
- IFW_SELECTOR_BLOCK_LNK. This table maps a block to a selector detail or to another block. This table is for future use.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_NOSP

The DAT_NOSP module provides data for mapping network source and destinations to new values for the FCT_NOSP module.

See the following documents:

- Identifying the Network Operator/Service Provider
- FCT_NOSP

Dependencies

DAT_NOSP module supports both file and database option.

If configured to get data from the database, the DAT_NOSP module requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-29 lists the DAT_NOSP registry entries.

Table 82-29 DAT_NOSP Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the database connection to the Pipeline Manager database.	Yes, if the data is stored in the database. Otherwise is not used.
FileName	Specifies the path and file name of the initialization file. See "Creating an NO/SP Data File". You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise is not used.
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	Yes
Source	Specifies where the data is stored: • **File** • **Database**	Yes

Sample Registry Entry for the Database Interface

```
NospData
{
  ModuleName = DAT_NOSP
  Module
  {
    Source = Database
    DataConnection = ifw.DataPool.Login
    ReuseOnFailure = FALSE
  }
}
```

Sample Registry Entry for the File Interface

```
NOSP
{
  ModuleName = DAT_NOSP
  Module
  {
    ReuseOnFailure = FALSE
    Source = File
    FileName = ./cfg/NOSP_Config1.dat
  }
}
```

Format of the file:

```
RANK;OLD_SOURCE;OLD_DESTINATION;A_PREFIX;NEW_SOURCE;NEW_DESTINATION;
```

For example,

```
ALL_RATE;1;ABC;BCD;0987;XYZ;YZA;
```

Semaphore File Entries

Table 82-30 lists the DAT_NOSP Semaphore file entries.

Table 82-30 DAT_NOSP Semaphore File Entries

Entry	Description
FileName	Specifies the path and file name of the initialization file.
Reload	Reloads data from the Pipeline Manager database. Only used if data is stored in the database.

Sample Semaphore File Entry for the Database Interface

```
ifw.DataPool.NOSP.Module.Reload {}
```

Sample Semaphore File Entry for the File Interface

```
ifw.DataPool.NOSP.Module.FileName = ./cfg/NOSP_Config2.dat
```

Database Tables

The DAT_NOSP module uses the following tables:

- IFW_NOSP
- IFW_GROUP

DAT_NumberPortability

The DAT_NumberPortability module provides number portability data to the FCT_NumberPortability module.

Registry Entries

Table 82-31 lists the DAT_NumberPortability registry entries.

Table 82-31 DAT_NumberPortability Registry Entries

Entry	Description	Mandatory
CountryCode	Specifies the country code, for example **49** for Germany. This is needed for normalization of CLIs. See "Configuring Normalization for Number Portability".	Yes
FileName	Specifies the name of the number portability file. See "Creating a Number Portability Data File".	Yes
InternationalAccessCode	Specifies the international access code. This is needed for normalization of CLIs. Default = 00 See "Configuring Normalization for Number Portability".	No

Table 82-31 (Cont.) DAT_NumberPortability Registry Entries

Entry	Description	Mandatory
InternationalAccessCodeSign	Specifies the international access code sign. This is needed for normalization of CLIs. Default = + See "Configuring Normalization for Number Portability".	No
NationalAccessCode	Specifies the national access code, for example **0** for Germany. This is needed for normalization of CLIs. See "Configuring Normalization for Number Portability".	Yes
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	No
SearchMethod	Specifies which search method to use. • **0** specifies to use best match. • **1** specifies to use exact match. • **2** specifies to use first prefix match. Default = **0** See "Configuring Number Portability Search".	No

Sample Registry Entry

```
NumberPortabilityData
{
  ModuleName = DAT_NumberPortability
  Module
  {
    FileName = ./data/primary_np.data
    CountryCode = 49
    NationalAccessCode = 0
    SearchMethod = 0
  }
}
```

Semaphore File Entries

Table 82-32 lists the DAT_NumberPortability Semaphore file entries.

Table 82-32 DAT_NumberPortability Semaphore File Entries

Entry	Description
AdditionalNumPortData	Specifies the name of the ASCII file that contains the newly ported numbers to reload. **Important:** This parameter must not be used with the Reload semaphore entry. Otherwise, an error message is logged and nothing is updated.
PrintData	Specifies the name of the ASCII file in which to print all newly ported numbers. Important: If this entry is specified with the **Reload** or **AdditionalNumPortData** semaphore entries, the **PrintData** entry is processed last.

Table 82-32 (Cont.) DAT_NumberPortability Semaphore File Entries

Entry	Description
Reload	Reloads data from the number portability data file. Important: You cannot use this entry at the same time as the **AdditionalNumPortData** semaphore entry.

Sample Semaphore File Entry

```
ifw.DataPool.NumberPortability.Module.Reload {}
ifw.DataPool.NumberPortability.Module.AdditionalNumPortData=./data/primary_np.data
ifw.DataPool.NumberPortability.Module.PrintData=./data/records.nport.dump
```

Events

Table 82-33 lists the DAT_NumberPortability Events.

Table 82-33 DAT_NumberPortability Events

Event Name	Trigger	Sender	Parameter
EVT_ADD_NUM_PORT_DATA_SUCCESSFUL	Adding new number Portability data succeeded.	DAT_NumberPortability	None
EVT_ADD_NUM_PORT_DATA_FAILED	Adding new number portability data failed.	DAT_NumberPortability	None

DAT_PortalConfig

The DAT_PortalConfig module loads data from the **/config/event_order_criteria**, **/config/business_params**, and **/config/credit_profile** objects in the BRM database.

Dependencies

This module requires a connection to the Database Connect (DBC) module. See "Database Connect (DBC)".

> **Note:**
>
> Due to the dependency of other data modules on DAT_PortalConfig, the DAT_PortalConfig registry entries must appear before all other data module entries in the registry file.

Registry Entries

Table 82-34 lists the DAT_PortalConfig registry entries.

Table 82-34 DAT_PortalConfig Registry Entries

Entry	Description	Mandatory
InfranetConnection	Specifies the connection to the DBC module.	Yes

Sample Registry Entry

```
PortalConfigDataModule
{
  ModuleName       = DAT_PortalConfig
  Module
  {
    InfranetConnection = ifw.DataPool.LoginInfranet
  }
}
```

Semaphore File Entries

Table 82-35 lists the DAT_PortalConfig Semaphore file entries.

Table 82-35 DAT_PortalConfig Semaphore File Entries

Entry	Description
CBPPrintData	Prints the /config/business_params data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data into a file named **DefaultCBPDataFile**_*Timestamp*.**lst**.
CreditProfilePrintData	Prints the **/config/credit_profile** data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data into a file named **DefaultConfigCreditProfileDataFile**_*Timestamp*.**lst**.
CBPReload	Reloads /config/business_params data.
CreditProfileReload	Reloads **/config/credit_profile** data.
PrintData	Prints all the data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data to the standard output console.
OODReload	Reloads **/config/event_order_criteria** data.
OODPrintData	Prints the **/config/event_order_criteria** data stored in the DAT_PortalConfig module's memory. If a filename is not provided, the module dumps the data into a file named **DefaultOODDataFile**_*Timestamp*.**lst**.

Sample Semaphore File Entry

```
ifw.DataPool.PortalConfigDataModule.Module.CreditProfile.Reload{}
ifw.DataPool.PortalConfigDataModule.Module.Reload{}
ifw.DataPool.PortalConfig.Module.CBPPrintData=BRM/config/prntCBPdata
ifw.DataPool.PortalConfig.Module.OODPrintData=BRM/config/prntOODdata
```

Events

Table 82-36 lists the DAT_PortalConfig events.

Table 82-36 DAT_PortalConfig Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Reload was successful.	DAT_PortalConfig	None
EVT_RELOAD_FAILED	Reload failed.	DAT_PortalConfig	None

Database Tables

The DAT_PortalConfig module uses the following database tables:

- CONFIG_T
- CONFIG_EVENT_ORDER_CRITERIA_T

DAT_PrefixDesc

The DAT_PrefixDesc module provides data for mapping phone number prefixes to descriptions, used by the FCT_PrefixDesc module.

See the following documents:

- Creating Call Destination Descriptions
- FCT_PrefixDesc

Dependencies

If data is stored in the database, the DAT_PrefixDesc module requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-37 lists the DAT_PrefixDesc registry entries.

Table 82-37 DAT_PrefixDesc Registry Entries

Entry	Description	Mandatory
CLIBase	Specifies if the zone tree values should be hexadecimal or decimal. You can use this entry in a semaphore file.	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes, if the data is stored in the database. Otherwise is not used.
PrefixDesc.File	Specifies the file prefix for the files that contain prefix descriptions. See "Creating a Prefix/Description Data File". You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise is not used.

Table 82-37 (Cont.) DAT_PrefixDesc Registry Entries

Entry	Description	Mandatory
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails. If **True**, the old data is used. If the entry is not used, the default is **False**.	Yes
Source	Specifies where the module gets data: • **File** • **Database**	Yes

Sample Registry Entry for the Database Interface

```
PrefixDescDataModule
{
  ModuleName = DAT_PrefixDesc
  Module
  {
    Source = Database
    DataConnection = ifw.DataPool.Login
    ReuseOnFailure = false
    CLIBase = 10
  }
}
```

Sample Registry Entry for the File Interface

```
PrefixDescData
{
  ModuleName = DAT_PrefixDesc
  Module
  {
    Source = File
    ReuseOnFailure = false
    CLIBase = 10
    PrefixDesc
    {
      File = ../daten/forgn_names.dat
      File = ../daten/onkz_names.dat
    }
  }
}
```

Semaphore File Entries

Table 82-38 lists the DAT_PrefixDesc Semaphore file entries.

Table 82-38 DAT_PrefixDesc Semaphore File Entries

Entry	Description
CLIBase	Specifies if the zone tree values should be hexadecimal or decimal. Valid values are 10(DEC) and 16(HEX).
PrefixDesc.File	Specifies the file prefix for the files that contain prefix descriptions.
Reload	Reloads the data.

Sample Semaphore File Entry

```
ifw.DataPool.PrefixDescDataModule.Module.Reload {}
```

Events

Table 82-39 lists the DAT_PrefixDesc events.

Table 82-39 DAT_PrefixDesc Events

Event name	Trigger	Sender	Parameter
EVT_RELOAD_FAILED	Update semaphore	DAT_PrefixDesc	None
EVT_RELOAD_SUCCESSFUL	Update semaphore	DAT_PrefixDesc	None

Database Tables

The DAT_PrefixDesc module uses the IFW_DESTINDESC database table.

See "Creating Call Destination Descriptions".

DAT_PriceModel

The DAT_PriceModel module provides pricing data for the FCT_MainRating module.

See the following documents:

- About Pipeline Rating
- FCT_MainRating

Dependencies

Requires a connection to the Pipeline Manager database.

This module uses event notification to refresh customized charge offer data. You must configure a connection to "DAT_Listener" if you plan to use this feature.

This module uses the **TailormadeProductsSearch** business parameter to skip lock on pricing. If you do not use tailor-made charge offers, and intend to skip lock on pricing, configure a connection to the DAT_PortalConfig module.

Registry Entries

Table 82-40 lists the DAT_PriceModel registry entries.

Table 82-40 DAT_PriceModel Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
Listener	Specifies the connection to the DAT_Listener module.	No

Table 82-40 (Cont.) DAT_PriceModel Registry Entries

Entry	Description	Mandatory
LogEvents	Specifies whether notification events received by the module are written to the process log file. Default = **False**	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module and enables DAT_PriceModel to retrieve business parameter settings from the DAT_PortalConfig module.	No

Sample Registry Entry

```
PriceModel
{
  ModuleName = DAT_PriceModel
  Module
  {
    DataConnection = ifw.DataPool.Login
    Listener = ifw.DataPool.Listener
  }
}
```

Semaphore File Entries

Table 82-41 lists the DAT_PriceModel Semaphore file entries.

Table 82-41 DAT_PriceModel Semaphore File Entries

Entry	Description
Reload	Reloads the data from the database.
PrintAllPriceModels	Prints all pricings in the configuration.
PrintOnePriceModel <PriceModel ID>	Prints the pricing ID.
PrintRangeOfPriceModels <PriceModel fromID> <PriceModel toID>	Prints all the pricings where ID is in the range.

Sample Semaphore File Entry

```
ifw.DataPool.PriceDataModule.Module.Reload {}
```

Events

Table 82-42 lists the DAT_PriceModel events.

Table 82-42 DAT_PriceModel Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Reload was successful.	DAT_PriceModel	None
EVT_RELOAD_FAILED	Reload failed.	DAT_PriceModel	None

Database Tables

The DAT_PriceModel module uses the following database tables:

- IFW_PRICEMODEL. This table stores pricing data.
- IFW_PRICEMDL_STEP. This table stores pricing step data.
- IFW_RESOURCE. This table stores resource configuration data.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_Rateplan

The DAT_Rateplan module provides charge data for the FCT_MainRating module.

Dependencies

Requires a connection to the Pipeline Manager database.

This module uses event notification to refresh customized charge offer data. You must configure a connection to "DAT_Listener" if you plan to use this feature.

This module uses the **TailormadeProductsSearch** business parameter to skip lock on charges. If you do not use tailor-made charge offers, and intend to skip lock on charges, configure a connection to the DAT_PortalConfig module.

Registry Entries

Table 82-43 lists the DAT_Rateplan registry entries.

Table 82-43 DAT_Rateplan Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
Listener	Specifies the connection to the DAT_Listener module.	No
LogEvents	Specifies whether notification events received by the module are written to the process log file. Default = **False**	No
PortalConfigDataModule	Specifies the connection to the DAT_PortalConfig module and enables DAT_RatePlan to retrieve business parameter settings from the DAT_PortalConfig module.	No
RowFetchSize	Specifies the number of rows of data to retrieve from the BRM database. Default = **1000**	**RowFetchSize**

Sample Registry Entry

```
Rateplan
{
  ModuleName = DAT_Rateplan
  Module
  {
    DataConnection = ifw.DataPool.Login
    Listener = ifw.DataPool.Listener
  }
}
```

Semaphore File Entries

Table 82-44 lists the DAT_Rateplan Semaphore file entries.

Table 82-44 DAT_Rateplan Semaphore File Entries

Entry	Description
Reload	Reloads the rating configuration data.
PrintAllRateplans	Prints all charges in the configuration.
PrintOneRateplan <RatePlan ID>	Prints the charge ID.
PrintRangeOfRateplans <RatePlan fromID> <RatePlan toID>	Prints all the charges where ID is in the range.

Sample Semaphore File Entry

```
ifw.DataPool.RateplanDataModule.Module.Reload {}
```

Events

Table 82-45 lists the DAT_Rateplan events.

Table 82-45 DAT_Rateplan Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Reload was successful.	DAT_Rateplan	None
EVT_RELOAD_FAILED	Reload failed.	DAT_Rateplan	None

Database Tables

The DAT_Rateplan module uses charge data from the following database tables:

- IFW_RATEPLAN
- IFW_RATEPLAN_VER
- IFW_RATEPLAN_CNF

The DAT_Rateplan module uses RUM data from the following database tables:

- IFW_RUM

- IFW_RUMGROUP
- IFW_RUMGROUP_LNK

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_Recycle

The DAT_Recycle module is used by standard recycling and Suspense Manager EDR to recycle EDRs. It connects to the DAT_Listener module and waits for business events that call for EDRs to be recycled.

This module creates a parameter file that allows the "EXT_InEasyDB" module to read suspended usage records associated with a recycle job. It also provides an interface for the "INP_Recycle" module to provide status updates about the EDR stream.

Dependencies

Requires a connection to the "DAT_Listener" module.

Registry Entries

Table 82-46 lists the DAT_Recycle registry entries.

Table 82-46 DAT_Recycle Registry Entries

Entry	Description	Mandatory
ControlPath	Specifies the path for SQL, parameter, job and restart files. **./database/Oracle/Scripts/Suspense**	Yes
Listener	Specifies the connection to the DAT_Listener module.	Yes
LogEvents	Specifies whether notification events received by the module are written to the process log file. Default = **False**	Yes
ParameterFile	Specifies the name of the parameter file which contains optional key/value entries.	Yes
ProcessCount	Specifies the threshold job count in the **QueueFileName** file. When the threshold is reached, the DAT_Recycle cleans up the queue. If not specified, the default value is **50**.	No
QueueFileName	Specifies the name of the file that stores recycle job IDs to be processed.	Yes
QueueFilePath	The path to the queue file specified for **QueueFileName**.	Yes

Sample Registry Entry

```
#------------------------------------------------------------
# Recycling Data
#------------------------------------------------------------
RecyclingData
```

```
{
  ModuleName                = DAT_Recycle
  Module
  {
    Listener = ifw.DataPool.Listener
    LogEvents = True
    ControlPath = ./database/Oracle/Scripts/Suspense
    ParameterFile = parameter.isc
    QueueFileName = RecycleJobIds_wireless.dat
    QueueFilePath = ./data
    ProcessCount = 50
  }
}
```

Semaphore File Entries

DAT_Recycle does not support semaphore updates.

DAT_ResubmitBatch

The DAT_ResubmitBatch module supports batch suspension and resubmission.

DAT_ResubmitBatch subscribes to "DAT_Listener" for ResubmitBatchRequest event. Upon receiving this, it gets information for all the batches corresponding to this ResubmitBatchRequest from the BRM database. It then moves all these batches to their respective pipeline input directories.

A ResubmitBatchRequest is propagated to the ifw through ifw_sync when a user resubmits a suspended batch with the SMC. During resubmission, a notification event (**/event/notification/suspense/batch_resubmit**) is generated with the admin action job id. This notification event is propagated to the ifw through ifw_sync in form of ResubmitBatchRequest.

Dependencies

This module requires connections to the following:

- BRM database.
- Pipeline Manager database.
- DAT_Listener module. See "DAT_Listener".

Registry Entries

Table 82-47 lists the DAT_ResubmitBatch registry entries.

Table 82-47 DAT_ResubmitBatch Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
Listener	Specifies the connection to the DAT_Listener module.	Yes
LogEvents	Logs each request received from the listener when true. Default = **True**	No

Table 82-47 (Cont.) DAT_ResubmitBatch Registry Entries

Entry	Description	Mandatory
QueueFileName	Name of file for file based queue. For example: **QueueFileName** = **ResubmitJobIds.dat**	No
QueueFilePath	Path of file for file based queue	No
QueueFileCleanupThreshold	For already processed events cleanup.	No
PipelineCategory	The Pipeline Category for the records. The module should only process records of its own pipeline category For example: **PipelineCategory= CDRPipeline**	Yes
TempDirectoryPath	Temporary directory path, used as a staging directory for resubmitted batches. It should not be used for any other purpose. For example: TempDirectoryPath = **./data/tmp**	No

Sample Registry Entry

```
ResubmitBatch
  {
    ModuleName = DAT_AccountBatch
    Module
    {
      DataConnection       = ifw.DataPool.LoginInfranet
      Listener             = ifw.DataPool.Listener
      LogEvents            = True
      QueueFileName        = ResubmitJobIds.dat
      QueueFilePath        = ./dataQueue
      FileCleanupThreshold = 50
      PipelineCategory     = CDRPipeline
      TmpDirectoryPath     = ./data/tmp
    }
  }
```

Semaphore File Entries

DAT_ResubmitBatch does not support semaphore updates.

DAT_ScenarioReader

The DAT_ScenarioReader module provides aggregation scenario data for the FCT_AggreGate module.

See the following documents:

- Setting Up Pipeline Aggregation
- FCT_AggreGate

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-48 lists the DAT_ScenarioReader registry entries.

Table 82-48 DAT_ScenarioReader Registry Entries

Entry	Description	Mandatory
Calendar	Specifies the calendar that is used for special day evaluation. Default = No calendar	No
DataCollection	Specifies a connection to the Pipeline Manager database.	Yes

Sample Registry Entry

```
ScenarioReader
{
  ModuleName = DAT_ScenarioReader
  Module
  {
    DataConnection = ifw.DataPool.Database
    Calendar = 2
  }
}
```

Semaphore File Entries

Table 82-49 lists the DAT_ScenarioReader Semaphore file entries.

Table 82-49 DAT_ScenarioReader Semaphore File Entries

Entry	Description
Reload	Reloads aggregation scenarios.

Sample Semaphore File Entry

```
ifw.DataPool.ScenarioReader.Module.Reload {}
```

Messages and Requests

Table 82-50 lists the DAT_ScenarioReader messages and requests.

Table 82-50 DAT_ScenarioReader Messages and Requests

Message/Request	Description	Sending/Receiving
REQ_EVENTHANDLER_NAME	Get the Event Handler.	Send to controller.

Events

Table 82-51 lists the DAT_ScenarioReader events.

Table 82-51 DAT_ScenarioReader Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_ScenarioReader	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_ScenarioReader	None

Database Tables

The DAT_ScenarioReader module uses the following database tables:

- IFW_SCENARIO. This table stores the aggregation scenario parameters. Some values can be overwritten by using the FCT_AggreGate registry.
- IFW_EDRC_FIELD. This table defines the EDR container fields for the aggregation scenarios. Each scenario uses exactly one EDR container description.
- IFW_CONDITION. This table stores the conditions that exclude an EDR or parts of an EDR from the aggregation process.
- IFW_GROUPING. This tables stores the scenario groupings that group aggregated results into subgroups. You can summarize the values within a grouping into subclasses.
- IFW_AGGREGATION. This tables stores aggregation functions and specifies how to handle the results.
- IFW_GROUPING_CNF. This table links data classes to a grouping.
- IFW_CLASS. This table defines the grouping classes. Each class consists of several class items.
- IFW_CLASSITEM. This table defines class items. All grouping values matching a class item are summarized and the class item code is added to the result.
- IFW_CLASS_LNK. This table links items and classes.
- IFW_CLASSCON. This table defines the class conditions. They determine which class to use when more than one class is associated to a grouping. A class condition specifies the dependency between a class from one grouping and a class item from another grouping.
- IFW_CLASSCON_LNK. This table links class conditions to one or more class items.

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_TimeModel

The DAT_TimeModel module provides time model, time zone, and day code data for the FCT_Mainrating module.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-52 lists the DAT_TimeModel registry entries.

Table 82-52 DAT_TimeModel Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies a connection to the Pipeline Manager database.	Yes

Sample Registry Entry

```
TimeModel
{
  ModuleName = DAT_TimeModel
  Module
  {
    DataConnection = ifw.DataPool.Login
  }
}
```

Semaphore File Entries

Table 82-53 lists the DAT_TimeModel Semaphore file entries.

Table 82-53 DAT_TimeModel Semaphore File Entries

Entry	Description
Reload	Reloads data from the Pipeline Manager database.

Sample Semaphore File Entry

```
ifw.DataPool.TimeDataModule.Module.Reload {}
```

Events

Table 82-54 lists the DAT_TimeModel events.

Table 82-54 DAT_TimeModel Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_TimeModel	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_TimeModel	None

Database Tables

The DAT_TimeModel module uses the following database tables:

- IFW_TIMEMODEL
- IFW_TIMEMODEL_LNK
- IFW_DAYCODE
- IFW_TIMEINTERVAL
- IFW_TIMEZONE

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_USC_Map

The DAT_USC_Map module provides usage scenario mapping data.

The DAT_USC_Map module retrieves mapping data from an ASCII file or from the Pipeline Manager database. This data is used by the FCT_USC_Map module to perform usage scenario mapping. See "FCT_USC_Map".

Dependencies

If the usage scenario mapping is stored in the database, this module requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-55 lists the DAT_USC_Map registry entries.

Table 82-55 DAT_USC_Map Registry Entries

Entry	Description	Mandatory
DataConnection	Specifies a connection to the Pipeline Manager database.	Yes, if the data is stored in the database. Otherwise not used.
LoadZoneDescription	Specifies whether to load the zone description into memory. Default = **False**	No
LoadZoneEntryName	Specifies whether to load the zone name into memory. Default = **False**	Yes
OptimizeFor	Specifies whether mapping should be optimized for **Speed** (the default) or **Memory**.	No
Source	Specifies where the USC mapping data is stored. The possible values are a **File** or **Database**.	Yes

Table 82-55 (Cont.) DAT_USC_Map Registry Entries

Entry	Description	Mandatory
USCMapFile	If **Source = File**, specifies file name and path that contains the USC mapping data. You can use this entry in a semaphore file.	Yes, if the data is stored in a file. Otherwise not used.
PreCompiledDataDir	Compiles USC mapping data and saves the data to the specified directory. Default = ./**compiled_usc_data** Files are stored in the format *USCzoneModelName*.**pc**. Make sure that the directory exists under the specified path. The compiled files are created in the first run of the pipeline. Before each subsequent run, they are validated and recompiled if necessary.	No
NumberOfThreads	Specifies the number of threads to use when loading and saving the precompiled mapping data. Default = **1**	No
UscGroups	Specifies the USC groups for which to load rules. Enclose the values in curly braces. For example: `UscGroups {TEL TEL_ROAMING TEL_INTL}` The default is to load all USC groups in the system. Use the semaphore when mapping rules are stored in the database (Source = **Database**).	No

Sample Registry Entry

```
USCDataModule
{
  ModuleName = DAT_USC_Map
  Module
  {
     Source               = DataBase
     DataConnection       = ifw.DataPool.Login
     LoadZoneDescription  = False
     LoadZoneEntryName    = False
     OptimizeFor          = MEMORY
     PreCompiledDataDir   = ./compiled_usc_data
     NumberOfThreads      = 5
     UscGroups            = {TEL TEL_ROAMING}
   }
}
```

Semaphore File Entries

Table 82-56 lists the DAT_USC_Map Semaphore file entries.

Table 82-56 DAT_USC_Map Semaphore File Entries

Entry	Description
LoadZoneDescription	Specifies whether to load the zone description into memory. Default = **False**

Table 82-56 (Cont.) DAT_USC_Map Semaphore File Entries

Entry	Description
LoadZoneEntryName	Specifies whether to load the zone name into memory. Valid values are **True** and **False**.
PrintAllUscMapData	Prints all the USC map data.
PrintUscMapDataForZoneModel	Prints the data for a given zone model ID.
PreCompiledDataDir	Compiles USC mapping data and saves the data to the specified directory. Default = **./compiled_usc_data** Files are stored in the format *USCzoneModelName*.**pc**. Make sure that the directory exists under the specified path. The compiled files are created in the first run of the pipeline. Before each subsequent run, they are validated and recompiled if necessary.
NumberOfThreads	Specifies the number of threads to use when loading and saving the precompiled mapping data. Default = **1**
UscGroups	Specifies the USC groups for which to load rules. If not set, all USC groups are loaded.
Reload	Command used to reload data into memory from the database.
USCMapFile	If **Source = File**, specifies file name and path that contains the USC mapping data.

Sample Semaphore File Entry

```
ifw.DataPool.USCDataModule.Module.UscGroups {TEL TEL_ROAMING}
ifw.DataPool.USCDataModule.Module.Reload {}
```

Database Tables

If the mapping data is stored in the Pipeline Manager database, The DAT_USC_Map module uses the IFW_USC_MAP database table. This table stores mapping rules for usage scenario maps.

For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

DAT_Zone

The DAT_Zone module provides zone data for the FCT_MainRating and the FCT_Zone modules. This module stores the real-time service class to Pipeline Manager service code mapping information in memory. When it processes realtime data, it returns the service code for a given service class.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 82-57 lists the DAT_Zone registry entries.

Table 82-57 DAT_Zone Registry Entries

Entry	Description	Mandatory
CliMode	Specifies if the zoning tree should be created in decimal (DEC) or hexadecimal (HEX) mode. Default = **DEC**	No
DataConnection	Specifies a connection to the Pipeline Manager database.	Yes, if the module gets data from the database.
DistCalcMode	Specifies the mode for calculating the distance between two area codes: • **DISC**. The configured coordinates are Cartesian coordinates. The distance is calculated using the Pythagorean theorem. • **GLOBE**. The coordinates are global. The distance is calculated using slower goniometric functions. You can use this entry in a semaphore file. Default = **DISC**	No
FileName	Specifies the path and file name for the zone model master file, if the module gets data from a file. You can use this entry in a semaphore file.	Yes, if the module gets data from a file
GeoFileName	Specifies the path and file name for the area code coordinate link file, if the module gets data from a file. You can use this entry in a semaphore file.	Yes, if the module gets data from a file
LoadZoneDescription	Specifies whether to load the zone descriptions into memory. Default = **False** You can use this entry in a semaphore file.	No
LoadZoneEntryName	Specifies whether to load the zone names into memory. Default = **False** You can use this entry in a semaphore file.	No
MaxAge	Specifies the maximum age of zone entries. If the value is **0** or null, all zone entries are loaded. You can use this entry in a semaphore file. Default = **0**	No
RealTime	Specifies whether the module should process real-time events. If **True**, the module processes real-time events, if **False**, the module processes batch events.	Yes
ReuseOnFailure	Specifies if the module should continue to use the old data if the **Reload** command fails: **True** = use the old data. **False** = do not use the old data. Default = **False**	Yes
Source	Specifies whether the module gets data from a file or the database.	Yes

Table 82-57 (Cont.) DAT_Zone Registry Entries

Entry	Description	Mandatory
ZoneModels	Specifies the source of the zone model codes: • If the source is a file, contains a list of zone model codes with the corresponding path and file name for the configuration file. • If the source is the database, contains a list of zone model codes that shall be used. You can use this entry in a semaphore file.	No, if the module gets data from the database.

Sample Registry for the Database Interface

```
Module
{
  ReuseOnFailure = FALSE
  MaxAge = 0
  Source = Database
  DataConnection = ifw.DataPool.Login
  LoadZoneDescription = False
  LoadZoneDescription = False
  ZoneModels
  {
    BASIC
    PROFI
    SPECIAL
  }
}
```

Sample Registry for the File Interface

Standard zone:

```
Module
{
   ReuseOnFailure = FALSE
   MaxAge = 90
   Source = File
   FileName = ./cfg/ZoneModelConfig.dat
   ZoneModels
   {
      ZM_ADD = /data9/INTEGRATE/TEST/config/ZM_ADD.dat
   }
}
```

Geographical zone:

```
Module
{
   ReuseOnFailure = FALSE
   MaxAge = 90
   Source = File
   FileName = ./cfg/ZoneModelConfig.dat
   GeoFileName = ./cfg/GeoAreaLink.dat
   ZoneModels
   {
      ZM_GEO = /data9/INTEGRATE/TEST/config/ZM_GEO.dat
```

```
    }
}
```

Sample Registry for Real-Time Zoning

```
Module
{
  ReuseOnFailure = FALSE
  Source = DataBase
  MaxAge = 0
  DistCalcMode = DISC
  DataConnection = ifw.DataPool.Login
  LoadZoneDescription = False
  LoadZoneEntryName = False
  RealTime = True
  ZoneModels
  {

  }
}
```

Semaphore File Entries

Table 82-58 lists the DAT_Zone Semaphore file entries.

Table 82-58 DAT_Zone Semaphore File Entries

Entry	Description
DistCalcMode	Specifies the mode for calculating the distance between two area codes: • **DISC**. The configured coordinates are cartesian coordinates. The distance is calculated using the Pythagorean theorem. • **GLOBE**. The coordinates are global. The distance is calculated using slower goniometric functions.
FileName	Specifies the path and file name for the zone model master file, if the module gets data from a file.
GeoFileName	Specifies the path and file name for the area code coordinate link file, if the module gets data from a file.
LoadZoneDescription	Specifies whether to load the zone descriptions into memory. **Note:** When this entry is updated through a semaphore, the reload semaphore must also be passed to reload the zone descriptions.
LoadZoneEntryName	Specifies whether to load the zone names into memory. **Note:** When this entry is updated through a semaphore, the reload semaphore must also be passed to reload the zone names.
MaxAge	Specifies the maximum age of zone entries. If the value is 0 or null, all zone entries are loaded.
Reload	Reloads the zoning data.
ZoneModels	Specifies the source of the zone model codes: • If the source is a file, contains a list of zone model codes with the corresponding path and file name for the configuration file. • If the source is the database, contains a list of zone model codes that shall be used.

Sample Semaphore File Entry for the Database Interface

```
ifw.DataPool.ZoneDataModule.Module.ZoneModels.BASIC.Reload {}
ifw.DataPool.ZoneDataModule.Module.ZoneModels.SPECIAL.Reload {}
```

Sample Semaphore File Entry for the File Interface

```
ifw.DataPool.ZoneDataModule.Module.ZoneModels.
ZM_ADD = /data9/INTEGRATE/test/config/ZM_ADD-new.dat

ifw.DataPool.ZoneDataModule.Module.ZoneModels.
ZM_MOBILE = /data9/INTEGRATE/test/config/ZM_MOBILE-new.dat
```

Events

Table 82-59 lists the DAT_Zone events.

Table 82-59 DAT_Zone Events

Event Name	Trigger	Sender	Parameter
EVT_RELOAD_SUCCESSFUL	Data reload was successful.	DAT_Zone	None
EVT_RELOAD_FAILED	Data reload failed.	DAT_Zone	None

Database Tables

The DAT_Zone module uses data from the following database tables:

- IFW_ZONEMODEL
- IFW_IMPACT_CAT
- IFE_STANDARD_ZONE
- IFW_GEO_MODEL
- IFW_GEO_ZONE
- IFW_GEOAREA_LNK

> **Note:**
>
> For information on compare patterns used in database values, see "About Using Regular Expressions when Specifying the Data to Extract".

83
Pipeline Manager iRules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager iRules.

IRL_EventTypeSplitting

The IRL_EventTypeSplitting iRule sends EDRs to separate output streams based on service codes.

See "Sending EDRs to Pipeline Output Streams".

Dependencies

This module must run after the FCT_ServiceCodeMap module and before the FCT_Reject module.

This is typically the last module before the FCT_Reject module.

For more information, see "Function Module Dependencies".

Sample Registry

```
EventSplitting
{
 ModuleName = FCT_IRules
 Module
 {
    Active = True
    Source = Database
    DataConnection = ifw.DataPool.DataConnection
    Rules {}
 }
}
```

EDR Container Fields

Table 83-1 lists the IRL_EventTypeSplitting EDR Container fields.

Table 83-1 IRL_EventTypeSplitting EDR Container Fields

Alias field name Default field name	Type	Access	Description
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Internal service code.

IRL_EventTypeSplitting_tt

The IRL_EventTypeSplitting_tt iRule sends EDRs to separate output streams.

See "Sending EDRs to Pipeline Output Streams".

Dependencies

This module must run after the FCT_ServiceCodeMap module and before the FCT_Reject module.

This is typically the last module before the FCT_Reject module.

For more information, see "Function Module Dependencies".

Sample Registry

```
ServiceOutputSplit
{
 ModuleName = FCT_IRules
 Module
 {
    Active = True
    Source = Files
    Rules
    {
    }
    Descriptions
    {
        ServiceCodeSplit = ./iScriptLib/iScriptLib_Standard/IRL_EventTypeSplitting_tt.irl
    }
 }
}
```

EDR Container Fields

Table 83-2 lists the IRL_EventTypeSplitting_tt EDR Container fields.

Table 83-2 IRL_EventTypeSplitting_tt EDR Container Fields

Alias field name Default field name	Type	Access	Description
LOGICAL_PARTITION_ID DETAIL.LOGICAL_PARTITION_ID	String	Read	Logical Partition ID.
INTERN_SERVICE_CODE DETAIL.INTERN_SERVICE_CODE	String	Read	Internal service code.

IRL_LeastCostPerEDR

The IRL_LeastCostPerEDR iRule flags all EDRs that satisfy the criteria for BRM least=1 cost rating. For more information, see "About Least Cost Rating".

You set up the criteria that an EDR must meet to qualify for least=1 cost rating in the **IRL_LeastCostPerEDR.irl** and **IRL_LeastCostPerEDR.data** files. See "Specifying the Rules to Qualify for Least Cost Rating".

Dependencies

This module must run:

- Before the "FCT_CustomerRating" module, because FCT_CustomerRating uses the flag set by this module to decide whether to create charge packets for all charge offers.
- After the "FCT_Filter_Set" module, because the rules you set up in **IRL_LeastCostRating.data** frequently use filter sets as one criteria for least=1 cost rating.

For more information, see "Function Module Dependencies".

Registry Entries

Table 83-3 lists the IRL_LeastCostEDR registry entries.

Table 83-3 IRL_LeastCostEDR Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. • **True** = Active • **False** = Inactive	Yes
LeastCostCheck	Specifies the path to the **IRL_LeastCostPerEDR.irl** file. See "Specifying the Rules to Qualify for Least Cost Rating".	Yes
Source	Specifies whether the least= cost rating data is stored in a file or in a database table. • **File** = The iRules data is stored in a file. • **Database** = The iRules data is stored in a database table.	Yes

Sample Registry

```
LeastCostPerEDR
{
   ModuleName = FCT_IRules
   Module
   {
      Active = False
      Source = Database
      DataConnection = ifw.DataPool.DataConnection
              Rules {}
   }
   }
```

EDR Container Fields

Table 83-4 lists the IRL_LeastCostEDR EDR Container fields.

Table 83-4 IRL_LeastCostEDR EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.LEAST_COST	Integer	Write	Toggles least= cost rating on and off. A value of **1** means do not apply least cost rating; a value of **2** specifies to apply least= cost rating.
DETAIL.CUST_A.MARKET_SEGMENT	String	Read	Specifies the filter set associated with the EDR.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all charge offers that are associated with the same service and event.
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Write	Contains the index of the highest priority rating charge offer. This is the charge offer with the highest rate priority for an event. In the case of two charge offers with matching priorities, the charge offer with the first start time is selected.

IRL_PipelineSplitting

The IRL_PipelineSplitting iRule is used in the pre-recycling pipeline to send EDRs to different output streams depending on their original pipeline names. The EDRs are then routed to their original pipelines for recycling.

The **PipelineSplitting.irl** file specified in the registry references a data file called **PipelineSplitting.data**, which you must modify based on your pipeline names. The default contents of the file are:

```
ALL_RATE;PreRecycleOutput
ALL_RATE_2;PreRecycleOutput_2
.*;PreRecylceOutput
```

See "Configuring a Pre-Recycling Pipeline".

Sample Registry

```
PipelineSplit
{
  ModuleName = FCT_IRules
  Module
  {
    Active = True
      Source = Database
      DataConnection = ifw.DataPool.DataConnection
      Rules
      {
      }
  }
}
```

IRL_PromotionalSavingPerEDR

The IRL_PromotionalSavingPerEDR iRule flags all EDRs that satisfy the criteria for a promotional savings calculation.

You set up the rules to qualify for the promotional savings calculation in the promotional savings iRules files (**IRL_PormotionalSavingPerEDR.irl** and **IRL_PromotionalSavingPerEDR.data**).

For more information, see "About Calculating the Promotional Savings".

Dependencies

This module must run before the IRL_LeastCost module and the FCT_CustomerRating module.

For more information, see "Function Module Dependencies".

Registry Entries

Table 83-5 lists the IRL_PromotionalSavingPerEDR registry entries.

Table 83-5 IRL_PromotionalSavingPerEDR Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. • **True** = Active • **False** = Inactive	Yes
PromotionalSaving	Specifies the path to the **IRL_PromotionalSavingPerEDR.irl** file. See "Specifying the Rules to Qualify for Promotional Savings".	No
Source	Specifies whether the iRules data is stored in a file or database table. • **File** = The iRules data is stored in a file. • **Database** = The iRules data is stored in a database table.	Yes

Sample Registry

```
PromotionalSavingPerEDR
{
   ModuleName = FCT_IRules
   Module
   {
      Active = True
      Source = Database
      DataConnection = ifw.DataPool.DataConnection
          Rules {}
   }
}
```

EDR Container Fields

Table 83-6 lists the IRL_PromotionalSavingPerEDR EDR Container fields.

Table 83-6 IRL_PromotionalSavingPerEDR EDR Container Fields.

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.PROMOTIONAL_SAVING	Integer	Write	Toggles promotional savings on and off. A value of **1** means do not apply promotional savings. A value of **2** applies promotional savings.
DETAIL.CUST_A.MARKET_SEGMENT	String	Read	Specifies the filter set associated with an EDR.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all charge offers that are associated with the same service and event.
DETAIL.CUST_A.INTER_RATING_PRODUCTS	String	Write	Contains the charge offer rating indexes. This is a comma-separated list of all rating charge offers' indexes associated with the same service and event, and their priorities.

IRL_UsageType

The IRL_UsageType iRule assigns usage types to EDRs.

Dependencies

This module must be run after the FCT_Account module and before the FCT_USC_Map module.

For more information, see "Function Module Dependencies".

Sample Registry

```
IRules
{
  ModuleName = FCT_IRules
  Module
  {
     Active = True
     Source = Database
     DataConnection = integrate.DataPool.DataConnection
     Rules {}
  }
}
```

EDR Container Fields

Table 83-7 lists the IRL_UsageType EDR Container fields.

Table 83-7 IRL_UsageType EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.ERA.PROFILE	String	Read	Contains the profile for customer A
DETAIL.CUST_B.ERA.PROFILE	String	Read	Contains the profile for customer B
DETAIL.CUST_A.PRODUCT.ERA.PROFILE	String	Read	Contains the charge offer profile for customer A
DETAIL.CUST_B.PRODUCT.ERA.PROFILE	String	Read	Contains the charge offer profile for customer B
DETAIL.CUST_A.ERA.PA.KEY	String	Read	Contains the customer A profile attribute key
DETAIL.CUST_A.ERA.PA.VALUE	String	Read	Contains the customer A profile attribute value
DETAIL.CUST_B.ERA.PA.KEY	String	Read	Contains the customer B profile attribute key
DETAIL.CUST_B.ERA.PA.VALUE	String	Read	Contains the customer B profile attribute value
DETAIL.CUST_A.PRODUCT.ERA.PA.KEY	String	Read	Contains the customer A charge offer profile attribute key
DETAIL.CUST_A.PRODUCT.ERA.PA.VALUE	String	Read	Contains the customer A charge offer profile attribute key
DETAIL.CUST_B.PRODUCT.ERA.PA.KEY	String	Read	Contains the customer B charge offer profile attribute key
DETAIL.CUST_B.PRODUCT.ERA.PA.VALUE	String	Read	Contains the customer B charge offer profile attribute key
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	String	Read	Contains the internal found purchases charge offer index.
DETAIL.USAGE_DIRECTION	String	Read	Contains the usage direction.
DETAIL.CONNECT_SUB_TYPE	String	Read	Contains the connection subtype
DETAIL.CUST_A.ACCOUNT_NO	String	Read	Contains the customer A account number.
DETAIL.CUST_B.ACCOUNT_NO	String	Read	Contains the customer B account number
DETAIL.CUST_A.SYSTEM_BRAND	String	Read	Contains the system brand for customer A
DETAIL.CUST_B.SYSTEM_BRAND	String	Read	Contains the system brand for customer B
DETAIL.CUST_A.BILL_CYCLE	String	Read	Contains the customer A bill cycle
DETAIL.B_NUMBER	String	Read	Contains the B number for call
DETAIL.ASS_GSMW_EXT.CELL_ID	String	Read	Contains the cell ID for GSM call
DETAIL.CHARGING_START_TIMESTAMP	String	Read	Contains the start time for call
DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Read	Contains the charge for customer A charge offer
DETAIL.CUST_B.PRODUCT.RATEPLAN_NAME	String	Read	Contains the charge for customer B charge offer

iRuleValidation

iRuleValidation is an instance of the FCT_IRules module used to validate the data in individual CIBER fields in the EDR container. iRuleValidation uses the **CIBER_VAL.xml** file that specifies the rules and rule items for validating CIBER fields.

You must load the rules in the **CIBER_VAL.xml** file into the Pipeline Manager database before starting Pipeline Manager.

Dependencies

Run this iRule before the ISC_TapSplitting module.

For more information , see "Function Module Dependencies".

Sample Registry

```
iRuleValidation
{
  ModuleName = FCT_IRules
  Module
  {
     Active = True
     Source = Database
     DataConnection = ifw.DataPool.DataConnection
     Rules
     {
        CIBER_VAL
     }
  }
}
```

84

Pipeline Manager iScripts

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager iScripts.

ISC_AddCBD

The ISC_AddCBD iScript prepares EDRs for rerating in the backout pipeline.

> **Note:**
>
> The ISC_AddCBD iScript is a deprecated module but remains in BRM for backward compatibility.

Dependencies

This module runs in its own backout pipeline for rerating.

For more information, see "Function Module Dependencies".

Sample Registry

```
AddCBD
{
  ModuleName = FCT_IScript
  Module
  {
    Active = TRUE
    Source = FILE
    Scripts
    {
      AddCBD
      {
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_AddCBD.isc
      }
    }

}
```

Modified Output Container Fields

The ISC_AddCBD iScript creates one associated charge breakdown record of type 981 with charge packets of type 680.

EDR Container Fields

Table 84-1 lists the ISC_AddCBD EDR container fields.

Table 84-1 ISC_AddCBD EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_PIN.ACCOUNT_POID	String	Read	Contains the BRM account POID.
DETAIL.ASS_PIN.BP.PIN_INFO_STRING	String	Read	Contains the string that stores aggregated balance packet information.
DETAIL.ASS_PIN.BP.PIN_AMOUNT	Decimal	Read/Write	Contains the monetary balance impact.
DETAIL.ASS_PIN.BP.PIN_DISCOUNT	Decimal	Read/Write	Contains the discount value
DETAIL.ASS_CBD.ACCOUNT_CODE	String	Write	Contains the account code related to charge packet being constructed.
DETAIL.ASS_CBD.RECORD_TYPE	String	Write	Contains the record type of charge packet.
DETAIL.ASS_CBD.CP	Data Block	Write	Charge packet data block.
DETAIL.DISCOUNT_KEY	String	Write	Contains the discount key.
DETAIL.ASS_CBD.CP.DP	Data Block	Write	Charge breakdown discount packet data block.

Required Input EDR Container Fields

The associated BRM billing record type 900 and the balance impacts of type 600 must be present.

ISC_ApplyTax

The ISC_ApplyTax iScript is used in the reprice pipeline during Incollect processing. This iScript retrieves the taxation flag value for the specific network operator from the in-memory cache. If the taxation flag is set to **on**, the tax amount is passed to the subscriber; otherwise, the tax amount is ignored.

Dependencies

This iScript requires the DAT_InterConnect module and the iScript extension IXT_OpFlag, which provide network operator configuration information that is accessed by ISC_ApplyTax.

For more information, see "Function Module Dependencies".

Sample Registry

```
ApplyTaxIScript
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         ApplyTaxIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ApplyTax.isc
```

```
        }
      }
   }
}
```

EDR Container Fields

Table 84-2 lists the ISC_ApplyTax EDR container fields.

Table 84-2 ISC_ApplyTax EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	Read/Write	Contains the charge amount value.
DETAIL.ASS_ROAMING_EXT.SENDER	String	Read	Contains the sender PLMN of the Transferred Account Procedure (TAP) file.

ISC_BACKOUTTypeSplitting

The ISC_BACKOUTTypeSplitting iScript is used by the backout pipeline for backout-only rerating. It determines if the EDRs are flagged for backout-only rerating and sends the EDRs to different output streams based on the event types.

Sample Registry

```
BackOutputSplit
{
     ModuleName = FCT_IScript
     Module
     {
          Active = TRUE
          Source = File
          Scripts
          {
               SplitScript
               {
                    FileName = ./iScriptLib/iScriptLib_Standard/ISC_BACKOUTTypeSplitting.isc
               }
          }
     }
}
```

ISC_CiberInputValidation

The ISC_CiberInputValidation iScript performs record-level validations of CIBER records.

Dependencies

Because erroneous CIBER records can be discarded, this module must run before the FCT_Discard module.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_CiberInputValidation
{
  ModuleName = FCT_IScript
  Module
  {
     Active = TRUE
     Source = FILE
     Scripts
     {
       CiberInputValidation
       {
            FileName = ./iScriptLib/iScriptLib_Standard/ISC_CiberInputValidation.isc
       }
    }
  }
}
```

ISC_CiberOutputMapping

The ISC_CiberOutputMapping iScript adds charge data to the ASSOCIATED_CIBER_EXTENSION block of the EDR. If the EDR does not contain an ASSOCIATED_CIBER_EXTENSION block, this iScript adds one.

Dependencies

This module must run after the FCT_MainRating module and the ISC_PostRating iScript.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_CiberOutputMapping
{
  ModuleName = FCT_IScript
  Module
  {
    Active = TRUE
    Source = FILE
    Scripts
    {
       CiberOutputMapping
       {
          AirRum = AIR
          TollRum = TOL
          OCCRum = OCC
          FileName = ./iScriptLib/iScriptLib_Standard/ISC_CiberOutputMapping.isc
       }
    }
  }
}
```

EDR Container Fields

Table 84-3 lists the ISC_CiberOutputMapping EDR container fields.

Table 84-3 ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	Read	Specifies the currency type.
CURRENCY_TYPE	String	Write	Specifies the currency type.
DETAIL.ASS_CBD.CP.RUM	String	Read	Checked to see if the EDR contains a toll charge.
TOLL_NETWORK_CARRIER_ID	String	Write	If the EDR contains a toll charge, this is set to **99001**, otherwise it is set to **00000**.
AIR_CHARGE Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE).	String	Write	If the value of AIR_CHARGABLE_TIME in the CIBER extension block is **0**, AIR_CHARGE is also set to **0**. Otherwise, it is set to the sum of the DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE in each charge packet for which all of the following statements are true: • RUM equals the value specified in the **AirRum** parameter in the registry • CHARGED_CURRENCY_TYPE equals "R" (rated) or "H" (home) • PRICEMODEL_TYPE equals "S" or "A" **Note:** If the value of DETAIL.ASS_CIBER_EXT.SPECIAL_FEATURES_USED in the CIBER extension block is F, then AIR_CHARGE is always set to **0.0**. The Special Features Used value is set in the CIBER input grammar.
AIR_RATE_PERIOD Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE).	String	Write	If the value of AIR_CHARGE is **0**, this field is set to **00**. Otherwise, this field is set to **01**. **Note:** If the value of DETAIL.ASS_CIBER_EXT.SPECIAL_FEATURES_USED in the CIBER extension block is F, then AIR_CHARGE is always set to **0.0**, and therefore AIR_RATE_PERIOD is set to **00**. The value of AIR_RATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **AirRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension AIR_RATE_PERIOD field is specific to your customization.

Table 84-3 (Cont.) ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
AIR_MULTIRATE_PERIOD Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE) .	N/A	Write	If the value of AIR_CHARGE is **0**, this field is set to **0**. Otherwise, this field is set to **1**. **Note:** If the value of DETAIL.ASS_CIBER_EXT.SPECIAL_FEATURES_USED in the CIBER extension block is F, then AIR_CHARGE is always set to **0.0**, and therefore AIR_MULTIRATE_PERIOD is set to **0**. The value of AIR_MULTIRATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **AirRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension AIR_RATE_PERIOD field is specific to your customization.
TOLL_CHARGE Set only for CIBER record type 22 (specified in DETAIL.ASS_CIBER_EXT.CIBER_RECORD_TYPE) .	N/A	Write	If the value of AIR_CHARGABLE_TIME in the CIBER extension block is **0**, TOLL_CHARGE is also set to **0**. Otherwise, it is set to the sum of the DETAIL.ASS_CBD.CHARGED_AMOUNT_VALUE in each charge packet for which all of the following statements are true: • RUM equals the value specified in the **AirRum** parameter in the registry • CHARGED_CURRENCY_TYPE equals "R" (rated) or "H" (home) • PRICEMODEL_TYPE equals "S" or "A"
TOLL_RATE_PERIOD (for CIBER record type 22 only)	N/A	Write	If the value of TOLL_CHARGE is **0**, this field is set to **00**. Otherwise, this field is set to **01**. The value of TOLL_RATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **TollRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension TOLL_RATE_PERIOD field are specific to your customization.

Table 84-3 (Cont.) ISC_CiberOutputMapping EDR Container Fields

Alias field name Default field name	Type	Access	Description
TOLL_MULTIRATE_PERIOD (for CIBER record type 22 only)	N/A	Write	If the value of TOLL_CHARGE is **0**, this field is set to **0**. Otherwise, this field is set to **1**. The value of TOLL_MULTIRATE_PERIOD is generally derived from the time interval code in the charge packet where RUM equals the value of the **TollRum** registry parameter. The time interval code is part of the time model that you define. Therefore, the fields used to derive the CIBER extension TOLL_MULTIRATE_PERIOD field are specific to your customization.
TOLL_RATE_PERIOD and TOLL_MULTIRATE_PERIOD (for CIBER record type 32 and 42 only) Sets these Charge NO 1 related fields: • **Charge NO 1** • **Charge NO 1 Indicator** • **Charge NO 1 Rate Period** • **Charge NO 1 Multi-Rate Period**	N/A	Write	**Charge NO 1** is set to the value of DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE. The values for the other Charge NO 1 related fields are based on the CIBER record type: Record type 32: • **Indicator** is set to **17** • **Rate Period** is set to **01** • **Multi-Rate Period** is set to **1** Record type 42: • **Indicator** is set to **12** • **Rate Period** is set to **04** • **Multi-Rate Period** is set to **1**

ISC_CiberRejectReason

The ISC_CiberRejectReason iScript sets a reason code in the CIBER extension block for records that are rejected for one of the following reasons:

- They are duplicates.
- There is no roaming agreement with the network operator.

Sample Registry

```
ISC_CiberRejectReason
{
  ModuleName = FCT_IScript
  Module
  {
     Active = TRUE
     Source = FILE
     Scripts
     {
        CiberRejectReason
        {
           FileName = ./iScriptLib/iScriptLib_CiberValidation/ISC_CiberRejectReason.isc
        }
```

```
      }
   }
}
```

ISC_ConsolidatedCP

The ISC_ConsolidatedCP iScript is used in the Incollect settlement pipeline and the Outcollect settlement pipeline. This iScript removes all non '00' impact category charge packets.

The TAP input grammar creates individual charge packets as well as consolidated charge packets. However, the FCT_BillingRecord module considers all charge packets for creating balance packets. For this reason, the individual charge packets (non '00' impact category charge packets) are removed and only consolidated charge packets are considered so that the balance amounts in the balance packets are correct.

This iScript also assigns the G/L ID to each consolidated charge packet based on the **GL_CODE** registry entry.

Registry Parameters

Table 84-4 lists the ISC_ConsolidatedCP registry parameter.

Table 84-4 ISC_ConsolidatedCP Registry Parameter

Registry Parameter	Description	Mandatory
GL_CODE	Specifies the G/L ID used for tracking the balance impacts of roaming usage events.	Yes

Sample Registry

```
ConsolidatedCP
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         ConsolidatedCPIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_ConsolidatedCP.isc
            GL_CODE = 1514
         }
      }
   }
}
```

EDR Container Fields

Table 84-5 lists the ISC_ConsolidatedCP EDR container fields.

Table 84-5 ISC_ConsolidatedCP EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Contains the impact category of the charge packet.
DETAIL.ASS_CBD.CP.RUM_ID	Long	Write	Contains the RUM ID.
DETAIL.ASS_CBD.CP.USAGE_GL_ACCOUNT_CODE	String	Write	Contains the G/L account.
DETAIL.ASS_CBD.TP.RELATED_CHARGE_INFO_ID	Integer	Write	Contains the corresponding charge packet index.

ISC_DupRAPRecords

The ISC_DupRAPRecords iScript is used in the RAP processing pipeline. It duplicates severe and fatal TAP records so that the records can be routed to multiple output streams.

Registry Parameters

Table 84-6 lists the ISC_DupRAPRecords registry parameter.

Table 84-6 ISC_DupRAPRecords Registry Parameter

Registry Parameter	Description	Mandatory
NULLStream	Specifies the DevNull output stream. See "OUT_DevNull".	Yes

Sample Registry

```
DupRAPRecords
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         DupRAPRecords
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_DupRAPRecords.isc
            NULLStream = DevNull
         }
      }
   }
}
```

EDR Container Fields

Table 84-7 lists the ISC_DupRAPRecords EDR container fields.

Table 84-7 ISC_DupRAPRecords EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ERROR_REJECT_TYPE	String	Write	FCT_Reject uses this to reject the detail to a stream other than the standard reject stream.
DETAIL.ASS_ROAMING_EXT.RAP_RECORD_TYPE	String	Read	RAP record type.

ISC_EDRToTAPOUTMap

The ISC_EDRToTAPOUTMap iScript is used to populate standard values to fields in the output TAP file based on its corresponding value in the EDR container. This is because for some fields EDR might have different internal representations and the TAP specification may ask for different representation for the same fields.

This iScript has to be customized by the user for mapping fields and specifying what is the internal EDR representation for the TAP fields specified value given in the standards document.

> **Note:**
>
> The ISC_EDRToTAPOUTMap iScript is a deprecated module but remains in BRM for backward compatibility.

Sample Registry

```
ISC_EDRToTAPOUTMap
{
  ModuleName = FCT_IScript
  Module
  {
     Active = TRUE
     Source = FILE
     Scripts
     {
       EDRToTAPOUTMap
       {
            FileName = ./iScriptLib/iScriptLib_Standard/ISC_EDRToTAPOUTMap.isc
       }
     }
  }
}
```

EDR Container Fields

Table 84-8 lists the ISC_EDRToTAPOUTMap EDR container fields.

Table 84-8 ISC_EDRToTAPOUTMap EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.DAY_CODE	String	Read / Write	External Day Code as estimated and used by the rating process. Sample mapping given in the script: (Can be customized as per user's requirement) WEEKDAY => N WEEKEND => P ALLDAYS => I
DETAIL.ASS_CBD.CP.TIME_INTERVAL_CODE	String	Read / Write	External Time Interval Code as estimated and used by the rating process. Sample mapping given in the script (Can be customized as per user's requirement): 20012 - O 20021 - I 20031 - P 20032 - O 20033 - O 20001 - P 0002 - S 20003 - S 20004 - S 20011 - P
DETAIL.ASS_CBD.CP.RUM	String	Read / Write	Classifies the charging part of a call in an intercarrier relationship, for interconnection or roaming. Sample mapping given in the script (Can be customized as per user's requirement): DUR - D SND - V REC - W EVT - E AIR - D OCC - F

ISC_FirstProductRealtime

This iScript sets the validity period of charge offers that start on first usage when they are used for the first time to rate an event in the real-time rerating pipeline.

> **Note:**
>
> To process first-usage events in the batch rating pipeline, use the FCT_FirstUsageNotify module.

Dependencies

Place this iScript in the real-time rerating pipeline before the FCT_DiscountAnalysis module.

For more information, see "Function Module Dependencies".

Sample Registry

```
FirstProductRealtime
{
    ModuleName = FCT_IScript
    Module
    {
        Active = True
        Source = File
        Scripts
        {
            FirstProductRealtime
            {
                 FileName = ./iScriptLib/iScriptLib_Standard/ISC_FirstProductRealtime.isc
            }
        }
    }
}
```

EDR Container Fields

Table 84-9 lists the ISC_FirstProductRealtime EDR container fields.

Table 84-9 ISC_FirstProductRealtime EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Decimal	Read	Contains an index of the customer's purchased charge offers that were used for rating.
DETAIL.CUST_A.PRODUCT.FIRST_USAGE_INDICATOR	Decimal	Read	Specifies whether the charge offer is configured to start when first used and the first-usage validity period status.
DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Specifies the customer account POID.
DETAIL.CUST_A.PRODUCT.OFFERING_POID	String	Read	Specifies the account's charge offer POID.
DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Specifies the charge offer's service POID.
DETAIL.CUST_A.PRODUCT.SERVICE_TYPE	String	Read	Specifies the charge offer's service type.
DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Specifies the EDR's start timestamp.
DETAIL.EVENT_ID	Decimal	Read	Specifies the event POID.
DETAIL.EVENT_TYPE	String	Read	Specifies the event type.
DETAIL.UTC_TIME_OFFSET	String	Read	Specifies the UTC time offset.

Table 84-9 (Cont.) ISC_FirstProductRealtime EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.REFRESH_BALANCE	Decimal	Write	Specifies whether the account's charge offer validity period has been updated in the BRM database. If set, the discount module retrieves the updated balance information before evaluating discounts for the event.

ISC_GetCamelFlag

The ISC_GetCamelFlag iScript is used in the roaming outcollect processing to retrieve the CAMEL flag information for a roaming partner. The CAMEL flag information is used during EDR processing.

Dependencies

The ISC_GetCamelFlag iScript must run before the FCT_PreRating module in the outcollect rating pipeline.

For more information, see "Function Module Dependencies".

Sample Registry

```
GetCamelFlag
{
     ModuleName = FCT_IScript
     Module
     {
          Active = True
          Source = FILE
          Scripts
          {
               CamelFlagIScript
               {
                    FileName =./iScriptLib/iScriptLib_Roaming/ISC_GetCamelFlag.isc
               }
          }
     }
}
```

ISC_GetResourceBalance

The ISC_GetResourceBalance iScript is used to get the memory balance of a balance element. This iScript returns either the balance amount or '0' if the balance retrieval is successfull. If a failure occurs, it returns '-1'.

The iScript should be in the same function pool and added after FCT_ApplyBalance in the batch pipeline.

Sample Registry

```
NewBalanceAPI
{
     ModuleName = FCT_IScript
     Module
     {
          Active = True
          Source = FILE
          Scripts
          {
               NewBalanceAPI
               {
                    FileName =./iScriptLib/iScriptLib_Standard/ISC_RetrieveBalance.isc
               }
          }
     }
}
```

ISC_LeastCost

The ISC_LeastCost iScript performs the following:

- Calculates and finds the lowest charge for an EDR. See "About Least Cost Rating".
- Calculates the total savings between the charge for a promotional charge offer and the charge for the lowest priority (base) charge offer. See "About Calculating the Promotional Savings".

Dependencies

This module must run after FCT_CustomerRating.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 84-10 lists the ISC_LeastCost registry parameters.

Table 84-10 ISC_LeastCost Registry Parameters

Registry Parameter	Description	Mandatory
Resource	Specifies the balance element type that this module uses to calculate any savings amount.	Yes
Resource_ID	Specifies the balance element IDs used to identify the balance element used when calculating the savings amount.	Yes

Sample Registry

```
FCT_LeastCostRating
{
    ModuleName = FCT_IScript
    Module
    {
```

```
            Active = True
            Source = File
            Scripts
            {
                myScript
                {
                    FileName = ./ISC_LeastCost.isc
                    Resource = "Saving Charge Resource"
                    Resource_ID = "1000100"
                }
            }
        }
    }
```

EDR Container Fields

Table 84-11 lists the ISC_LeastCost EDR container fields.

Table 84-11 ISC_LeastCost EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Integer	Read/write	The amount charged to the customer.
DETAIL.ASS_CBD.DP.AMOUNT	Integer	Read	The amount of discount between the amount charged to the customer, and a greater amount that could have been charged.
DETAIL.ASS_CBD.CP.RATEPLAN_CODE	String	Read	The code identifying the least cost rating charge.
DETAIL.CUST_A.PRODUCT.RATEPLAN_NAME	String	Read	A description of the least cost rating charge code.
DETAIL.CUST_A.PRODUCT_PRIORITY	String	Read	Contains a list of the priorities for all charge offers that are associated with the same service and event.
DETAIL.CUST_A.INTERN_RATING_PRODUCTS	String	Write	Contains the charge offer rating indexes. This is a comma-separated list of all rating charge offers' indexes associated with the same service and event, and their priorities.
DETAIL.ASS_CBD.CP.RESOURCE	String	Write	The balance element to impact for reporting promotional savings.
DETAIL.ASS_CBD.CP.RESOURCE_ID	Integer	Write	The ID of the balance element to impact for promotional savings.
DETAIL.ASS_CBD.CP.RESOURCE_TYPE	Integer	Write	The savings charge packet to impact. This is **992** by default.

ISC_MapNetworkOperatorInfo

The ISC_MapNetworkOperatorInfo iScript maps the DETAIL.SOURCE_NETWORK field to the PIN_FLD_ORIGIN_NETWORK field and the DETAIL.DESTINATION_NETWORK field to the PIN_FLD_DESTINATION_NETWORK field of the opcode input block for the corresponding event.

Dependencies

For more information, see "Function Module Dependencies".

Sample Registry

```
MapNetworkOperatorInfo
  {
   ModuleName = FCT_IScript
   Module
   {
     Active = TRUE
     Source = File
     Scripts
     {
       PreOpcode
       {
         FileName = ./iScriptLib/AAA/ISC_MapNetworkOperatorInfo.isc
       }
     }
   }
  }
```

ISC_Migration

Use the ISC_Migration iScript during account migration to automatically flag EDRs for suspension. The FCT_Suspense module can then route the EDRs to a separate suspense output stream.

Sample Registry

```
MigrationPlugIn
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = File
    Scripts
    {
      MigrationIScript
      {
        FileName = ./samples/wireless_splitter/ISC_Migration.isc
      }
    }
  }
}
```

ISC_MiscOutcollect

The ISC_MiscOutcollect iScript is used in the Outcollect rating pipeline. This module adds BASIC_SERVICE and SUPPLEMENTARY_SERVICE blocks to the EDR container for GSM services. This is done to ensure that the TAP file generated by the pipeline contains all the required information.

For GPRS services, this module adjusts the record number field in the GPRS extension block to **0**.

It also modifies the RUM field of the charge packets as follows:

- DUR is replaced by D.
- SND is replaced by V.
- REC is replaced by W.

Sample Registry

```
MiscAddInfo
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         MiscAddInfoIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_MiscOutCollect.isc
         }
      }
   }
}
```

EDR Container Fields

Table 84-12 lists the ISC_MiscOutcollect EDR container fields.

Table 84-12 ISC_MiscOutcollect EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_GSMW_EXT.BS_PACKET.RECORD_TYPE	String	Write	Contains the record type field.
DETAIL.ASS_GSMW_EXT.BS_PACKET.RECORD_NUMBER	Integer	Write	Contains the record number field.
DETAIL.ASS_GSMW_EXT.BS_PACKET.CHAIN_REFERENCE	String	Write	Contains the call reference.
DETAIL.ASS_GSMW_EXT.BS_PACKET.LONG_DURATION_INDICATOR	String	Write	Contains whether the call is a long duration or not.
DETAIL.ASS_GSMW_EXT.BS_PACKET.BASIC_SERVICE	String	Write	Contains the basic service of the call.
DETAIL.ASS_GSMW_EXT.BS_PACKET.QOS_REQUESTED	String	Write	Contains the QOS requested.
DETAIL.ASS_GSMW_EXT.BS_PACKET.QOS_USED	String	Write	Contains the QOS used.
DETAIL.ASS_GSMW_EXT.BS_PACKET.CHARGING_START_TIMESTAMP	Date	Write	Contains the call start time.

Table 84-12 (Cont.) ISC_MiscOutcollect EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_GSMW_EXT.BS_PACKET.CHARGING_END_TIMESTAMP	Date	Write	Contains the call end time.
DETAIL.ASS_GSMW_EXT.BS_PACKET.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	Write	Contains the total charged amount value of the call.
DETAIL.ASS_GSMW_EXT.BS_PACKET.WHOLESALE_CHARGED_TAX_RATE	Decimal	Write	Contains the tax rate.
DETAIL.ASS_GSMW_EXT.BS_PACKET.SPEECH_VERSION_REQUESTED	String	Write	Contains the speech version requested.
DETAIL.ASS_GSMW_EXT.BS_PACKET.SPEECH_VERSION_USED,	String	Write	Contains the speech version used.
DETAIL.ASS_GSMW_EXT.SS_PACKET.RECORD_TYPE	String	Write	Contains the record type.
DETAIL.ASS_GSMW_EXT.SS_PACKET.RECORD_NUMBER	Integer	Write	Contains the record number.
DETAIL.ASS_GSMW_EXT.SS_PACKET.SS_EVENT	String	Write	Contains the event type.
DETAIL.ASS_GSMW_EXT.SS_PACKET.ACTION_CODE	String	Write	Contains the connect type.
DETAIL.ASS_CBD.CP.RUM	String	Read/Write	Contains the ratable unit of measurement.
DETAIL.ASS_GPRS_EXT.RECORD_NUMBER	Integer	Write	Contains the record number.

ISC_Monitoring

The ISC_Monitoring iScript records latencies for authentication, authorization, and accounting requests.

Dependencies

The ISC_Monitoring iScript depends on the ISC_StartTime iScript.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 84-13 lists the ISC_Monitoring registry parameters.

Table 84-13 ISC_Monitoring Registry Parameters

Registry Parameter	Description	Mandatory
FileName	Specifies the location of the ISC_Monitoring iScript.	Yes
recordsPerFile	Specifies the number of records per event log file.	Yes
recordsPerWrite	Specifies the number of records per write operation.	Yes
eventLogDir	Specifies the directory for the event log file.	Yes

Sample Registry

```
MonitoringPlugIn
{
   ModuleName = FCT_IScript
   Module
     {
       Active = True
       Source = FILE
       Scripts
       {
          ReadIScript
          {
             FileName = ./iScriptLib/AAA/ISC_Monitoring.isc
             recordsPerFile = 1000 # number of records per event log file
             recordsPerWrite = 10 # number of records per write operation
             eventLogDir = ./log/dump # directory for series of event log file
          }
       }
   }
}
```

EDR Container Fields

Table 84-14 lists the ISC_Monitoring EDR container fields.

Table 84-14 ISC_Monitoring EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.MILLISEC_TIME	Integer	Read	Specifies the latency time in milliseconds.
DETAIL.OPCODE_NUM	Integer	Read	Number of the BRM opcode that performs the requested action.

ISC_NRTRDE_ErrorReport

The ISC_NRTRDE_ErrorReport iScript is used during roaming incollect processing by the NRTRDE processing pipeline. This iScript collects the validation errors in the EDRs and creates error records in the Pipeline Manager database. It also collects NRTRDE file processing information and creates file processing records in the Pipeline Manager database. Information stored in the validation and file processing records in the database are used for generating NRTRDE reports.

Dependencies

The ISC_NRTRDE_ErrorReport iScript must run after the ISC_NrtrdeHeaderValidation iScript.

For more information, see "Function Module Dependencies".

Registry Parameters

The registry parameter is **FileName**. It is mandatory. **FileName** specifies the location of the iScript. The default location is:

./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_ErrorReport.isc

Sample Registry

```
NRTRDE_ErrorReport
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
         NRTRDE_ErrorReport
        {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_ErrorReport.isc
            DatabaseConnection = ifw.DataPool.Login
        }
      }
   }
}
```

ISC_NRTRDE_EventSplit

The ISC_NRTRDE_EventSplit iScript is used by roaming outcollect processing to duplicate and route EDRs to the corresponding roaming partner's NRTRDE output streams based on the roaming partner's NRTRDE flag.

Dependencies

The ISC_NRTRDE_EventSplit iScript must run after the FCT_EnhancedSplitting module in the outcollect rating pipeline.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 84-15 lists the ISC_NRTRDE_EventSplit registry parameters.

Table 84-15 ISC_NRTRDE_EventSplit Registry Parameters

Registry Parameter	Description	Mandatory
FileName	Specifies the location of the ISC_NRTRDE_EventSplit iScript. The default location is: **./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_EventSplit.isc**	Yes
NRTRDE_STREAM_PATTERN	Specifies the name of the NRTRDE stream pattern. This parameter appends the PLMN to the NRTRDE stream pattern to retrieve the output stream name.	Yes

Sample Registry

```
NRTRDESplit
{
```

```
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         NRTRDESplit
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_NRTRDE_EventSplit.isc
            NRTRDE_STREAM_PATTERN = NRTRDEOutput_
         }
      }
    }
}
```

ISC_NrtrdeHeaderValidation_v2_01

The ISC_NrtrdeHeaderValidation_v2_01 iScript is used during roaming incollect processing by the NRTRDE processing pipeline. This iScript validates the information in the header record of the TD35 file based on the TD35 specifications.

Dependencies

The ISC_NrtrdeHeaderValidation_v2_01 iScript must run before any other modules in the NRTRDE processing pipeline.

For more information, see "Function Module Dependencies".

Registry Parameters

The registry parameter is **FileName**. It is mandatory. **FileName** specifies the location of the iScript. The default location is:

./iScriptLib/iScriptLib_Tap3Validation/ISC_NrtrdeHeaderValidation_v2_01.isc

Sample Registry

```
TapValidationIScripts
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
         NrtrdeHeaderValidation
         {
             FileName = ./iScriptLib/iScriptLib_Tap3Validation/ISC_NrtrdeHeaderValidation_v2_01.isc
         }
      }
   }
}
```

ISC_ObjectCacheTypeOutputSplitter

Use the ISC_ObjectCacheTypeOutputSplitter iScript to enter a value in an EDR to create two identical output files from a single input EDR and write them to separate output streams.

Dependencies

To use this iScript, you must have object cache residency distinction enabled in your system.

Requires a connection to the DAT_BalanceBatch module.

For more information, see "Function Module Dependencies".

Sample Registry

```
ObjectCacheTypeOutputSplitter_Script
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = File

    Scripts
    {
      ObjectCacheTypeOutputSplit
      {
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_ObjectCacheTypeOutputSplitter.isc
      }
    }
  }
}
```

ISC_OverrideRateTag

The ISC_OverrideRateTag iScript is used in the outcollect settlement pipelines to populate the RATE_TAG field with the value of the NRTRDE flag in the balance impact.

The value of the RATE_TAG field is used by high-usage reports to filter out NRTRDE operator data from the report.

Dependencies

The ISC_OverrideRateTag iScript must run after the FCT_BillingRecord module in the outcollect settlement pipeline.

For more information, see "Function Module Dependencies".

Sample Registry

```
OverrideRateTag
{
     ModuleName = FCT_IScript
     Module
     {
```

```
          Active = True
          Source = FILE
          Scripts
          {
               OverrideRateTagIScript
               {
                    FileName =./iScriptLib/iScriptLib_Roaming/ISC_OverrideRateTag.isc
               }
          }
     }
}
```

ISC_OverrideSuspenseParams

The ISC_OverrideSuspenseParams iScript overrides some fields in the Suspense Extension block of the EDR container that is set by Suspense Manager.

During Outcollect processing, this iScript in the RAP Processing pipeline overrides the PIPELINE_NAME suspense field. For severe TAP records, PIPELINE_NAME is set to the name of the outcollect rating pipeline. For fatal TAP records, PIPELINE_NAME is set to the Suspense Batch output stream.

Registry Parameters

Table 84-16 lists the ISC_OverrideSuspenseParams registry parameters.

Table 84-16 ISC_OverrideSuspenseParams Registry Parameters

Registry Parameter	Description	Mandatory
TAPFilePrefix	Specifies the prefix of the TAP file. • **CD** for files containing chargeable data • **TD** for files containing test data	Yes
TAPOutCollectPipeline	Specifies the name of the outcollect rating pipeline.	Yes
TAPCorrectionPipeline	Specifies the path of the directory from which the original TAP file was sent to the network operator.	Yes
TAPSentArchivePath	Specifies the path of the directory from which the original TAP file was sent to the network operator.	Yes
SBRStream	Specifies the output stream that generates the suspense batch file.	Yes

Sample Registry

```
OverrideSuspenseParams
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         OverrideSuspenseParams
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_OverrideSuspenseParams.isc
            TAPFilePrefix = CD
            TAPOutCollectPipeline = TAPOutCollectPipeline
```

```
            TAPCorrectionPipeline = TAPCorrectionPipeline

            #the following path must be absolute
            TAPSentArchivePath = ./data/outcollect/tapout/sent
            SBRStream = SBROutput
         }
      }
   }
}
```

EDR Container Fields

Table 84-17 lists the ISC_OverrideSuspenseParams EDR container fields.

Table 84-17 ISC_OverrideSuspenseParams EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_ROAMING_EXT.TAP_FILE_SEQ_NO	Integer	Read	Sequence number of the TAP file.
DETAIL.ASS_ROAMING_EXT.RAP_RECORD_TYPE	String	Read	RAP record type.
DETAIL.BATCH_ID	String	Write	Records the TAP batch ID.
DETAIL.ORIGINAL_BATCH_ID	String	Write	Records the TAP batch ID.
DETAIL.ASS_SUSPENSE_EXT.SUSPENDED_FROM_BATCH_ID	String	Write	Records the TAP batch ID.
DETAIL.ASS_SUSPENSE_EXT.SOURCE_FILENAME	String	Write	Records the TAP file name.
DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME	String	Write	For a severe error, records the TAP outcollect rating pipeline name. For a fatal error, records the TAP correction pipeline name.
DETAIL.ASS_SUSPENSE_EXT.RECYCLE_KEY	String	Write	Records the RAP file sequence number.
DETAIL.ASS_ROAMING_EXT.TAP_FILE_PATH	String	Write	The path of the directory from which the original TAP file was sent to the network operator (Only for fatal errors).
DETAIL.ASS_ROAMING_EXT.SUSPENSION_TIME	Date	Write	(Only for fatal errors) Suspension time of the TAP file.

ISC_PopulateOpcodeandUtilBlock_Diameter

The ISC_PopulateOpcodeandUtilBlock_Diameter iScript adds an opcode block in the EDR. In the Dispatcher pipeline, the EDR is duplicated to handle failover. The ISC_PopulateOpcodeandUtilBlock_Diameter iScript improves the performance by reducing the time taken to duplicate the EDR. Also, it provides the flexibility of performing minor validations. This iScript populates the opcode block and other relevant fields like OPCODE_NODE and OPCODE_NUMBER, which are required by the TimeoutRouter pipeline.

Dependencies

This iScript must be called before calling FCT_Opcode.

For more information, see "Function Module Dependencies".

Sample Registry

```
ProcessPipeline
  {
    PopulateOpcodeAndUtilBlock
  {
    ModuleName? = FCT_IScript
    Module
     {
       Active = TRUE
       Source = File
       Scripts
       {
         PreOpcode?
         {
           FileName? = ./iScriptLib/AAA/ISC_PopulateOpcodeAndUtilBlock.isc
      }
   }
}
```

ISC_PostRating

The ISC_PostRating iScript adds all the retail and wholesale charges and puts them in the DETAIL.RETAIL_CHARGED_AMOUNT_VALUE and DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE fields.

See "Billing Consolidation with CIBER Roaming and Revenue Assurance".

Dependencies

This module must run:

- After rating modules FCT_CustomerRating, FCT_PreRating, and FCT_MainRating

  or

- After the FCT_ExchangeRate module

For more information, see "Function Module Dependencies".

Sample Registry

```
PostRating
{
    ModuleName = FCT_IScript
    Module
    {
        Active = True
        Source = File

        Scripts
        {
           PostRating
           {
              FileName = ./ISC_PostRating.isc

              RetailRecordType = 981
              RetailResource = DEM
              RetailPricemodelType = S
```

```
                RetailCurrencyType = R

                WholesaleRecordType = 990
                WholesaleResource = DEM
                WholesalePricemodelType = S
                WholesaleCurrencyType = R
            }
        }
    }
}
```

EDR Container Fields

Table 84-18 lists the ISC_PostRating input EDR container fields.

Table 84-18 ISC_PostRating Input EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.RECORD_TYPE	String	N/A	Contains the charge breakdown record type (retail or wholesale).
DETAIL.ASS_CBD.CP.RESOURCE	String	N/A	Contains the charge breakdown balance element type.
DETAIL.ASS_CBD.CP.PRICEMODEL_TYPE	String	N/A	Contains the charge breakdown pricing.
DETAIL.ASS_CBD.CP.CHARGED_CURRENCY_TYPE	String	N/A	Contains the charge breakdown currency type for charged amount.
DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	N/A	Contains the charge breakdown impact category.
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_CURRENCY	String	N/A	Contains the charge breakdown currency type for charged amount.
DETAIL.ASS_CBD.CP.CHARGED_TAX_TREATMENT	String	N/A	Contains the charge breakdown tax treatment type.
DETAIL.ASS_CBD.CP.CHARGED_AMOUNT_VALUE	Decimal	N/A	Contains the charge breakdown charged amount.

Table 84-19 lists the ISC_PostRating output EDR container fields.

Table 84-19 ISC_PostRating Output EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.RETAIL_IMPACT_CATEGORY	String	N/A	Contains the retail impact category.
DETAIL.RETAIL_CHARGED_AMOUNT_VALUE	Decimal	N/A	Contains the retail charged amount value.
DETAIL.RETAIL_CHARGED_AMOUNT_CURRENCY	String	N/A	Contains the retail charged amount currency.
DETAIL.RETAIL_CHARGED_TAX_TREATMENT	String	N/A	Contains the retail charged amount tax treatment.
DETAIL.WHOLESALE_IMPACT_CATEGORY	String	N/A	Contains the wholesale impact category.
DETAIL.WHOLESALE_CHARGED_AMOUNT_VALUE	Decimal	N/A	Contains the wholesale charged amount value.
DETAIL.WHOLESALE_CHARGED_AMOUNT_CURRENCY	String	N/A	Contains the wholesale charged amount currency.

Table 84-19 (Cont.) ISC_PostRating Output EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.WHOLESALE_CHARGED_TAX_TREATMENT	String	N/A	Contains the wholesale charged amount tax treatment.

ISC_ProfileAnalyzer

For each EDR, the ISC_ProfileAnalyzer iScript compares the telephone numbers or other relevant data in EDRs with the ERAs that the customer's service owns. ISC_ProfileAnalyzer stores each ERA label that matches the relevant EDR container field.

The iScript adds the label names to the EDR container field DETAIL.PROFILE_LABEL_LIST. If there are multiple names, the names are separated by a comma by default. You can change the separator character in the registry. If a label name contains a comma, you need to change the separator character.

Dependencies

ISC_ProfileAnalyzer depends on the ERA values populated by FCT_Account. It must be run after FCT_Account and before any rating modules in the pipeline.

For more information, see "Function Module Dependencies".

Sample Registry

```
ProfileAnalyzer
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
        ProfileAnalyzer
        {
        ServiceType = TEL
        ProfileName = FRIENDS_FAMILY
        LabelSeparator = ,
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_ProfileAnalyzer.isc
        }
     }
   }
}
```

EDR Container Fields

The ISC_ProfileAnalyzer iScript uses the EDR container fields listed in Table 84-20:

Table 84-20 ISC_ProfileAnalyzer EDR Container Fields

Alias field name Default field name	Type	Access	Description
PROFILE_LABEL_LIST DETAIL. PROFILE_LABEL_LIST	String	Write	Contains unique profile labels when the profile attributes match an EDR container field used for comparison to find F&F list.
DETAIL.CUST_A.PRODUCT. ERA.LABEL	String	Read	Contains a label associated with the service: for example, MYFAMILY.
DETAIL.CUST_A.SHARED_PROFILE_LIST. ERA.LABEL	String	Read	Contains a label associated with the service profile from a shared profile.
ERA.PROFILE	String	Read	Contains the profile name: for example, "Friends&Family."
CUST_A. SHARED_PROFILE_LIST	Block	Read	Contain the profiles that the service owns or shares as a member of a profile sharing group.
CUST_B.SHARED_PROFILE_LIST	Block	Read	Contain the profiles that the service owns or shares as a member of a profile sharing group.
CUST_A. SHARED_PROFILE_LIST .ERA	Block	Read	Contains shared ERA information.
CUST_B.SHARED_PROFILE_LIST.ERA	Block	Read	N/A
ERA.NAME	String	Read	Contains the profile attribute name.
ERA.VALUE	String	Read	Contains the profile attribute value.

ISC_ProfileLabel

The ISC_ProfileLabel iScript is used when rating CDRs based on ERAs. It determines whether the ERA profiles specified in the **ProfileName** registry entry match the EDR field value and populates the DETAIL.PROFILE_LABEL_LIST field with the ERA labels of the matching ERAs, and the DETAIL.USAGE_TYPE field with appropriate usage type for the ERA.

Dependencies

The ISC_ProfileLabel iScript must run after the FCT_Account module and before any rating modules.

Requires a connection to DAT_AccountBatch. This iScript works only in batch rating.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 84-21 lists the ISC_ProfileLabel registry parameters.

Table 84-21 ISC_ProfileLabel Registry Parameters

Registry Parameter	Description	Mandatory
ProfileName	Specifies the name of the ERA profile to analyze.	Yes

Table 84-21 (Cont.) ISC_ProfileLabel Registry Parameters

Registry Parameter	Description	Mandatory
LabelSeparator	ERA labels in the DETAIL.PROFILE_LABEL_LIST EDR field are separated using this delimiter. The default is a comma (,).	No
FileName	Specifies the location of the ISC_ProfileLabel iScript. The default location is **./iScriptLib/iScriptLib_Standard/ISC_ProfileLabel.isc**.	Yes

Sample Registry

```
ProfileLabel
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      ProfileLabel
      {
      ProfileName = FRIENDS_FAMILY
      LabelSeparator = ,
      FileName = ./iScriptLib/iScriptLib_Standard/ISC_ProfileLabel.isc
      }
    }
  }
}
```

EDR Container Fields

Table 84-22 list the ISC_ProfileLabel EDR container fields.

Table 84-22 ISC_ProfileLabel EDR Container Fields

Alias field name Default field name	Type	Access	Description
CUST_A_INTERN_PP_INDEX DETAIL.CUST_A.INTERN_FOUND_PP_INDEX	Integer	Read	Contains an index of the customer's purchased charge offers identified by the FCT_Account module.
CUST_A.ACCOUNT_PARENT_ID DETAIL.CUST_A.ACCOUNT_PARENT_ID	String	Read	Account ID of the service for which usage is getting rated.
CUST_A.PRODUCT.SERVICE_ID DETAIL.CUST_A.PRODUCT.SERVICE_ID	String	Read	Service ID for which usage is getting rated.
B_NUMBER DETAIL.B_NUMBER	String	Read	Called number of the event.
BDR_CHARGING_START_TIMESTAMP DETAIL.CHARGING_START_TIMESTAMP	Date	Read	Contains the event's starting timestamp. The timezone information of this timestamp is stored in the BDR_UTC_TIME_OFFSET field.

Table 84-22 (Cont.) ISC_ProfileLabel EDR Container Fields

Alias field name Default field name	Type	Access	Description
BDR_UTC_TIME_OFFSET DETAIL.UTC_TIME_OFFSET	String	Read	Contains the UTC time offset that normalizes the charging start timestamp to the UTC tim zone. All validity timestamps in the BRM customer data are stored in normalized UTC time. The format is +/– *HHMM*.
PROFILE_LABEL_LIST DETAIL.PROFILE_LABEL_LIST	String	Write	Contains ERA labels that contain a value that matches the EDR field value.
USAGE_TYPE DETAIL.USAGE_TYPE	String	Write	Contains the usage type for the ERA.

ISC_RAP_0105_InMap

The ISC_RAP_0105_InMap iScript copies TAP data from staging fields in the EDR container to business fields in the EDR container. This iScript is used during roaming outcollect processing.

Dependencies

This should be the first module in the **FunctionPool**.

For more information, see "Function Module Dependencies".

Sample Registry

```
BusinessMapping
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       BusinessMapping
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_RAP_0105_InMap.isc
       }
     }
  }
}
```

ISC_RemoveASSCBD

The ISC_RemoveASSCBD iScript is used in the Outcollect rating pipeline to remove associated charge breakdown packets associated with RAP records that are recycled to the pipeline during RAP file processing.

Sample Registry

```
RemoveASSCBD
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         RemoveASSCBDIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_RemoveASSCBD.isc
         }
      }
   }
}
```

EDR Container Fields

Table 84-23 lists the ISC_RemoveASSCBD EDR container fields.

Table 84-23 ISC_RemoveASSCBD EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_CBD.RECORD_NUMBER	Integer	Read/Write	Contains the record number.

ISC_RollbackSettlement

During roaming incollect and outcollect settlement processing, the ISC_RollbackSettlement iScript checks for errors in the EDR. When there is an error, it notifies the Transaction Manager to roll back the transactions in the settlement pipeline. The Transaction Manager then notifies the FCT_BatchSuspense module to suspend the entire input file.

Sample Registry

```
RollbackSettlement
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         RollbackIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_RollbackSettlement.isc
         }
      }
   }
}
```

ISC_SetAndValidateBatchInfo

The ISC_SetAndValidateBatchInfo iScript populates and validates the batch related fields for the EDR container.

The iScript validates the HEADER.BATCH_ID. If it does not exist, the entire batch is rejected. If it exists, then it copies the HEADER.BATCH_ID to DETAIL.BATCH_ID.

If DETAIL.EVENT_ID is missing in an EDR, the EDR is rejected.

Dependencies

This iScript must be placed at the beginning of the pipeline so that the batch ID is inserted before any further processing of the mediation batches. It should be used only if you do not use Suspense Manager.

For more information, see "Function Module Dependencies".

Registry Entries

Table 84-24 lists the ISC_SetAndValidateBatchInfo registry entries.

Table 84-24 ISC_SetAndValidateBatchInfo Registry Entries

Entry	Description	Mandatory
ValidateOriginalBatchId	• If **True**, and if DETAIL.ORIGINAL_BATCH_ID is missing, the EDR is rejected. • If **False**, it copies the HEADER.BATCH_ID in DETAIL.ORIGINAL_BATCH_ID.	Yes
KeepBatchIds	• If **True**, this iScript does not modify the values in DETAIL.ORIGINAL_BATCH_ID and DETAIL.BATCH_ID. • If **False**, and if **ValidateOriginalBatchId** is **True**, this iScript assigns the values in HEADER.BATCH_ID to DETAIL.BATCH_ID. • If **False**, and if **ValidateOriginalBatchId** is **False**, this iScript assigns the value in HEADER.BATCH_ID to DETAIL.ORIGINAL_BATCH_ID and DETAIL.BATCH_ID.	Yes

Sample Registry

```
SetAndValidateBatchInfo
   {
    ModuleName = FCT_IScript
     Module
     {
       Active = True
       Source = FILE
       Scripts
        {
          SetAndValidateBatchInfo
        {
          FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetAndValidateBatchInfo.isc
          ValidateOriginalBatchId = TRUE
          KeepBatchIds = TRUE
        }
      }
```

```
    }
}
```

ISC_SetEDRStatus

The ISC_SetEDRStatus iScript sets the EDR status to **Success**, **Suspense**, **Duplicate**, **Discard**, or **Skipped** for each EDR.

Dependencies

This iScript must be used before FCT_AggreGate and before the scenario that collects audit data grouped on the EDRStatus field.

For more information, see "Function Module Dependencies".

Sample Registry

```
SetEDRStatus
  {
    ModuleName = FCT_IScript
     Module
      {
       Active = True
        Source = FILE
         Scripts
      {
        SetEDRStatus
      {
        FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetEDRStatus.isc
      }
    }
  }
}
```

ISC_SetOutputStream

The ISC_SetOutputStream iScript sets the Output Stream to **TelOut**, **SMSOut**, **GPRSOut**, **RejectOut**, or **DuplicateOut** for each EDR.

Dependencies

This iScript must be used after FCT_Reject, because it is dependent on the fields set in FCT_Reject, and before the scenario that collects audit data grouped on the OutputStream field.

For more information, see "Function Module Dependencies".

Sample Registry

```
SetOutputStream
  {
    ModuleName = FCT_IScript
    Module
   {
     Active = True
     Source = FILE
```

```
    Scripts
   {
     SetOutputStream
   {
      FileName = ./iScriptLib/iScriptLib_RevenueAssurance/ISC_SetOutputStream.isc
    }
   }
  }
}
```

ISC_SetRevenueFigures

The ISC_SetRevenueFigures iScript collects the previous and current charged and discount amount for a configured balance element ID.

Dependencies

This iScript must run after rating and discounting and before FCT_AggreGate.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 84-25 lists the ISC_SetRevenueFigures registry parameter.

Table 84-25 ISC_SetRevenueFigures Registry Parameter

Registry Parameter	Description	Mandatory
ResourceId	Specifies the balance element for which you want to calculate the discount value.	Yes

Sample Registry

```
SetRevenueFigures
{
 ModuleName = FCT_IScript
 Module
 {
  Active = True
  Source = FILE
  Scripts
  {
   SetRevenueFigures
   {
   FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetRevenueFigures.isc
   ResourceId = 978
   }
  }
 }
}
```

ISC_SetRevenueStream

The ISC_SetRevenueStream iScript sets the Revenue Stream to **Retail**, **Wholesale**, **Roaming**, or **Unknown** for each EDR.

Dependencies

This iScript must be used before FCT_AggreGate and after post rating (after the EDRs are rated).

For more information, see "Function Module Dependencies".

Sample Registry

```
SetRevenueStream
{
  ModuleName = FCT_IScript
  Module
 {
  Active = True
  Source = FILE
  Scripts
  {
   SetRevenueStream
   {
    FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetRevenueStream.isc
   }
  }
 }
}
```

ISC_SetSvcCodeRTZoning

For each EDR, the ISC_SetSvcCodeRTZoning iScript finds the service type and updates DETAIL.INTERN_SERVICE_CODE EDR field with the customized service code value.

Sample Registry

```
SetSvcCodeRTZoning
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = FILE
      Scripts
      {
         SetSvcCodeRTZoning
         {
            FileName = ./iScriptLib/iScriptLib_Standard/ISC_SetSvcCodeRTZoning.isc
         }
      }
   }
}
```

EDR Container Fields

Table 84-26 lists the ISC_SetSvcCodeRTZoning EDR container fields.

Table 84-26 ISC_SetSvcCodeRTZoning EDR Container Field

Alias field name Default field name	Type	Access	Description
DETAIL.INTERN_SERVICE_CODE	String	Write	Contains the service type of the current EDR.

ISC_StartTime

The ISC_StartTime iScript is used to request the start time.

Sample Registry

```
StartTime
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      StartTime
      {
      FileName = ./iScriptLib/iScriptLib_Standard/ISC_StartTime.isc
      }
    }
  }
}
```

EDR Container Fields

Table 84-27 lists the ISC_StartTime EDR container fields.

Table 84-27 ISC_StartTime EDR Container Field

Alias field name Default field name	Type	Access	Description
DETAIL.MILLISEC_TIME	Integer	Read	Specifies the latency time in milliseconds.

ISC_TapDetailValidation_v3_12

The ISC_TapDetailValidation_v3_12 iScript validates that the fields present in the detail record of the EDR container contain valid data.

> **Note:**
>
> The ISC_TapDetailValidation_v3_12 iScript is a deprecated module but remains in BRM for backward compatibility.

You run the ISC_TapDetailValidation_v3_12 iScript when incoming files that you receive from your roaming partner use the TAP format. When processing the incoming TAP files, BRM converts input data from TAP fields into corresponding fields of the EDR container description file.

Dependencies

Because an erroneous TAP record can be discarded before the record is split into multiple records, this module must run before the FCT_Discard and ISC_TapSplitting modules.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_TapDetailValidation_v3_12
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TapDetailValidation_v3_12
       {
           FileName = ./iScriptLib/iScriptLib/ISC_TapDetailValidation_v3_12.isc
       }
     }
  }
}
```

EDR Container Fields

Table 84-28 lists the ISC_TapDetailValidation_v3_12 EDR container fields.

Table 84-28 ISC_TapDetailValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
SERVER_TYPE_OF_NUMBER DETAIL.ASS_CAMEL_EXT.SERVER_ TYPE_OF_NUMBER	Integer	Read	CAMEL Invocation Fee TD57 item. Performs number-normalization.
CHARGEABLE_QUANTITY_VALUE DETAIL.ASS_CBD.CP.CHARGEABLE_ QUANTITY_VALUE	Decimal	Read	Chargeable Units TD57 item. Indicates the number of chargeable units. The value should be equal to or greater than zero.

Table 84-28 (Cont.) ISC_TapDetailValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
IMPACT_CATEGORY DETAIL.ASS_CBD.CP.IMPACT_CATEGORY	String	Read	Charge Type TD57 item. Identifies the type of charge. Possible values: • **00** Total charge for Charge Information (the invoiceable value) • **01** Airtime charge • **02** reserved • **03** Toll charge • **04** Directory assistance • **05** – **20** reserved • **21** VPMN surcharge • **69** – **99** reserved
TAX_TYPE DETAIL.ASS_CBD.TP.TAX_TYPE	String	Read	Tax Type TD57 item. Indicates the type of tax. Possible values: • **01** National • **02** Regional • **03** County • **04** Local/City
VOLUME_RECEIVED DETAIL.VOLUME_RECEIVED	Decimal	Read	Data Volume Incoming TD57 item. Identifies the number of incoming octets (bytes) within an occurrence of GPRS Service Used or Content Service Used. The value should be equal to or greater than zero.
VOLUME_SENT DETAIL.VOLUME_SENT	Decimal	Read	Data Volume Outgoing TD57 item. Identifies the number of outgoing octets (bytes) within an occurrence of GPRS Service Used or Content Service Used. The value should be equal to or greater than zero.
WHOLESALE_IMPACT_CATEGORY DETAIL.WHOLESALE_IMPACT_CATEGORY	String	Read	Charge Type TD57 item. Identifies the wholesale type of charge.

ISC_TapHeaderTrailerValidation_v3_12

The ISC_TapHeaderTrailerValidation_v3_12 iScript validates that fields present in the header and trailer records of the EDR contain valid data.

> **Note:**
>
> The ISC_TapHeaderTrailerValidation_v3_12 iScript is a deprecated module but remains in BRM for backward compatibility.

You run the ISC_TapHeaderTrailerValidation_v3_12 iScript when incoming files that you receive from your roaming partner use the TAP format. When processing the incoming TAP files, BRM converts input data from TAP fields into corresponding fields of the EDR container description file.

Dependencies

Because an erroneous TAP record can be discarded before the record is split into multiple records, this module must run before the FCT_Discard and ISC_TapSplitting modules.

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_TapHeaderTrailerValidation_v3_12
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TapHeaderTrailerValidation_v3_12
       {
            FileName = ./iScriptLib/iScriptLib/ISC_TapHeaderTrailerValidation_v3_12.isc
       }
     }
  }
}
```

EDR Container Fields

Table 84-29 lists the ISC_TapHeaderTrailerValidation_v3_12 EDR container fields.

Table 84-29 ISC_TapHeaderTrailerValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
CHARGE_REFUND_INDICATOR DETAIL.ASS_CONT_EXT.SERVICE_USED_ LIST.CHARGE_REFUND_INDICATOR	Integer	Read	Charge Refund Indicator TD57 item. Indicates that Content Transaction is a refund. When present, changes the signs of any revenue within Content Service Used. **Value:** **1** Refund
TOTAL_CHARGE_REFUND TRAILER.TOTAL_CHARGE_VALUE_ LIST.TOTAL_CHARGE_REFUND	String	Read	Total Charge Refund TD57 item. Specifies the sum of all the charges associated with Charge Type when Charge Type represents a refund. The value must be greater than zero.

Table 84-29 (Cont.) ISC_TapHeaderTrailerValidation_v3_12 EDR Container Fields

Alias field name Default field name	Type	Access	Description
TOTAL_NUMBER_OF_RECORDS TRAILER.TOTAL_NUMBER_OF_RECORDS	Integer	Read	Specifies the total number of Basic Records in the file, excluding header and trailer. Used as a check value to determine that all records have been correctly transmitted or used.

ISC_TapSplitting

The ISC_TapSplitting iScript splits mobile originating and terminating EDRs when the CDR contains more than one basic service. ISC_TapSplitting creates a new EDR for each additional basic service.

For information about splitting MOC and MTC records, see "Generating Multiple TAP MOC and MTC Records".

Dependencies

Must run after the following modules:

- FCT_DuplicateCheck
- FCT_Discard

For more information, see "Function Module Dependencies".

Sample Registry

```
ISC_TapSplitting
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TapSplitting
       {
            FileName = ./iScriptLib/iScriptLib/ISC_TapSplitting.isc
       }
      }
  }
}
```

ISC_TaxCalc

The ISC_TaxCalc iScript applies a flat tax to pipeline-rated events.

Dependencies

Run this module after the FCT_MainRating module, but before the FCT_BillingRecord module.

For more information, see "Function Module Dependencies".

Registry Parameters

Table 84-30 lists the ISC_TaxCalc registry parameters.

Table 84-30 ISC_TaxCalc Registry Parameters

Registry Parameter	Description	Mandatory
FlatTaxRate	Specifies the default flat tax percentage. Pipeline Manager applies this tax rate when an event does not match any criteria in the **taxcodes_map** file. For example, set **FlatTaxRate** to **5** to apply a 5% tax on the charged amount.	Yes
TaxCode	Specifies the default tax code. Pipeline Manager uses this tax code when an event does not match any criteria in the **taxcodes_map** file.	Yes
TaxCodeMapFilePath	Specifies the path to the **taxcodes_map** file. If the parameter is missing, the module retrieves tax code map data from the **/config/taxcodes_map** object.	No
TaxType	Specifies the default tax type. Pipeline Manager applies this tax type when an event does not match any criteria in the **taxcodes_map** file.	Yes

Sample Registry

```
TaxPlugin
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
         Tax
            FlatTaxRate = 5
            TaxCode = FLAT
            TaxType = 16
            TaxCodeMapFilePath = BRM_home/sys/cm/taxcodes_map
            Filename = ./iScriptLib/ISC_TaxationLib/ISC_TaxCalc.isc
     }
  }
}
```

Semaphore File Entries

Table 84-31 lists the ISC_TaxCalc Semaphore file entries.

Table 84-31 ISC_TaxCalc Semaphore File Entries

Entry	Description
TaxCodeMapReload	Reloads the tax code map data to pipeline memory.
TaxCodeMapPrintData	Prints the tax code map data to the console.
TaxCodeMapPrintData =*filename*	Prints the tax code map data into a log file, where *filename* is the name of the log file.

Sample Semaphore File Entry

```
ifw.DataPool.PortalConfigDataModule.Module.TaxCodeMapReload {}
ifw.DataPool.PortalConfigDataModule.Module.TaxCodeMapPrintData {}
ifw.DataPool.PortalConfigDataModule.Module.TaxCodeMapPrintData = taxCodeMapData.log
```

ISC_TAP_0312_Include

The ISC_TAP_0312_Include iScript copies TAP data from staging fields in the EDR container to business fields in the EDR container.

This iScript is part of the following iScripts:

- ISC_TAP_0312_InMap
- ISC_RAP_0105_InMap

Sample Registry

```
BusinessMapping
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       BusinessMapping
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_Include.isc
       }
     }
  }
}
```

EDR Container Fields

Table 84-32 lists the ISC_TAP_0312_Include EDR container fields.

Table 84-32 ISC_TAP_0312_Include EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.ASS_LTE_EXT.REQUESTEDNUMBER	String	Read	Contains the number (TEL URI of the original destination) to which the customer requested to be connected.
DETAIL.ASS_LTE_EXT.REQUESTEDPUBLICUSERID	String	Read	Contains the public user ID (SIP URI of the original destination) to which the customer requested to be connected.

ISC_TAP_0312_InMap

The ISC_TAP_0312_InMap iScript copies TAP data from staging fields in the EDR container to business fields in the EDR container.

Registry Entries

Table 84-33 lists the ISC_TAP_0312_inMap registry entries.

Table 84-33 ISC_TAP_0312_InMap Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive: **True** = Active **False** = Inactive	Yes
Source	Specifies the source of the iScripts: • **File** • **Database**	Yes
FileName	Specifies the location of the iScript. The default location is **./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_InMap.isc**.	Yes

Sample Registry

```
BusinessMapping
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       BusinessMapping
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_InMap.isc
       }
     }
  }
}
```

ISC_TAP_0312_Validations

The ISC_TAP_0312_Validations iScript validates TAP input data.

Registry Entries

Table 84-34 lists the ISC_TAP_0312_Validations registry entries.

Table 84-34 ISC_TAP_0312_Validations Registry Entries

Entry	Description	Mandatory
Active	Specifies if the module is active or inactive. **True** = Active **False** = Inactive	Yes
Source	Specifies the source of the iScripts. • **File** • **Database**	Yes
FileName	Specifies the location of the iScript. The default location is **./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_Validations.isc**.	Yes

Sample Registry

```
TAP3Validations
{
  ModuleName = FCT_IScript
  Module
  {
     Active = True
     Source = File
     Scripts
     {
       TAP3Validations
       {
           FileName = ./iScriptLib/iScriptLib_Roaming/ISC_TAP_0312_Validations.isc       }
     }
  }
}
```

ISC_UsageClassSetting

The ISC_UsageClassSetting iScript is used in the Incollect and Outcollect settlement pipelines to populate the DETAIL.USAGE_CLASS field with the value of DETAIL.CONNECT_TYPE in the EDR container, which specifies the type of call. The value of DETAIL.USAGE_CLASS is stored in the **event_dlay_sess_tlcs** table and is later used by high-usage reports to determine premium calls.

Sample Registry

```
UsageClassMap
{
   ModuleName = FCT_IScript
   Module
   {
      Active = True
      Source = File
      Scripts
      {
         UsageClassSettingIScript
         {
            FileName = ./iScriptLib/iScriptLib_Roaming/ISC_UsageClassSetting.isc
         }
      }
```

```
    }
}
```

EDR Container Fields

Table 84-35 lists the ISC_UsageClassSetting EDR container fields.

Table 84-35 ISC_UsageClassSetting EDR Container Fields

Alias field name Default field name	Type	Access	Description
DETAIL.USAGE_CLASS	String	Write	Internal usage class.
DETAIL.CONNECT_TYPE	String	Read	Contains the connect type of the call.

UpdateTapInfo_StopRapout

The UpdateTapInfo_StopRapout iScript updates the database with information on the Stop Return RAP file sent to the Visited Public Mobile Network (VPMN) operator.

Dependencies

None

Sample Registry

```
UpdateTapInfo_StopRapout
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      UpdateTapInfo_StopRapout
      {
        FileName = ./iScriptLib/iScriptLib_Roaming/ISC_UpdateTapInfo_StopRapout.isc
        DatabaseConnection = ifw.DataPool.Login
      }
    }
  }
}
```

UpdateTapInfo_Tapin

The UpdateTapInfo_Tapin iScript updates the information in the database on incoming TAP files for use in generating Stop Return RAP files.

Dependencies

This iScript should be configured before the FCT_Reject module, and after any iScripts that populate the mandatory fields in the header of an EDR.

For more information, see "Function Module Dependencies".

Sample Registry

```
UpdateTapInfo_Tapin
{
  ModuleName = FCT_IScript
  Module
  {
    Active = True
    Source = FILE
    Scripts
    {
      UpdateTapInfo_Tapin
      {
        FileName = ./iScriptLib/iScriptLib_Roaming/ISC_UpdateTapInfo_Tapin.isc
        DatabaseConnection = ifw.DataPool.Login
      }
    }
  }
}
```

85
Pipeline Manager iScript Functions

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager iScript functions.

Arithmetic Functions

Table 85-1 contains the arithmetic functions.

Table 85-1 Arithmetic Functions

Function	Description
decimalAbs	Derives an absolute value from a decimal value.
decimalToLong	Converts the integer portion of a decimal value to a Long value.
longAbs	Derives an absolute value from a Long value.
longToDecimal	Converts a Long value to a decimal value.
round	Rounds a decimal value to a specified number of decimal places.
sqrt	Calculates the square root of the input value.
trunc	Truncates a decimal value to a specified number of decimal places.

decimalAbs

This function derives an absolute value from a decimal value.

Syntax

```
Decimal decimalAbs(Decimal source);
```

Parameter

source
The decimal value from which to derive the absolute value.

Return Values

Returns the derived absolute value.

Example

```
if ( x == decimalAbs( x ) )
{
  logFormat( "x is a positive value" );
}
```

decimalToLong

This function converts the integer portion of a decimal value to a Long value.

Syntax

```
Long decimalToLong(Decimal source);
```

Parameter

source
The decimal value to convert to a Long value.

Return Values

Returns the Long value of the integer portion of the decimal value.

Example

```
Long p = decimalToLong( 3.1415 );
```

longAbs

This function derives an absolute value from a Long value.

Syntax

```
Long longAbs(Long source);
```

Parameter

source
The Long value from which to derive the absolute value.

Return Values

Returns the derived absolute value.

Example

```
if ( x == longAbs( x ) )
{
  logFormat( "x is a positive value" );
}
```

longToDecimal

This function converts a Long value to a decimal value.

Syntax

```
Decimal longToDecimal(Long value);
```

Parameter

value
The Long value to convert to a decimal value.

Return Values

Returns the converted decimal value.

Example

```
Decimal bytesPerSecond = longToDecimal( bytes ) / \
longToDecimal( seconds );
```

round

This function rounds a decimal value to a specified number of decimal places.

Syntax

```
Decimal round(Decimal value [, Long places] [, String mode]);
```

Parameters

value
The value to round.

places
The number of decimal places to achieve when rounding, also known as the number of significant digits (the default is **0**).

mode
The rounding mode, or method of rounding. Possible values:

- **ROUND_PLAIN:** (Default) If the digit following the last significant digit is 5 or greater, round up. If the digit following the last significant digit is less than 5, round down.
- **ROUND_UP:** Always round up if the digit following the last significant digit is greater than 0.
- **ROUND_DOWN:** Always round down. This is the same as truncating all digits following the last significant digit.
- **ROUND_BANKERS:** This mode rounds in one of the following ways, depending on the value of the digit following the last significant digit:
  - If it is less than 5, truncate all digits following the last significant digit.
  - If it is greater than 5, round up.
  - If it is 5, round to the nearest even digit. For example, if the precision is 2, 10.155 and 10.165 both round to 10.16 because 6 is an even number.

Return Values

Returns the value rounded to the specified decimal place.

Example

```
Decimal r = round( 3.1415, 3 ); // r now is 3.142
```

sqrt

This function calculates the square root of the input value.

Syntax

```
Decimal sqrt(Decimal value);
```

Parameter

value
The value for which to calculate the square root.

Return Values

Returns the square root of the input value.

Example

```
Decimal c = sqrt( a*a + b*b );
```

trunc

This function truncates a decimal value to a specified number of decimal places.

Syntax

```
Decimal trunc(Decimal value [, Long places]);
```

Parameters

value
The value to truncate.

places
The number of decimal places by which the value should be truncated (the default is **0**).

Return Values

Returns the value truncated to the specified decimal place.

Example

```
Decimal t = trunc( 3.1415, 3 ); // t now is 3.141
```

ASN.1 Functions

Table 85-2 contains the ASN.1 functions.

Table 85-2 ASN.1 Functions

Function	Description
asnTreeAddInteger	Adds an integer object to the current active node of the ASN tree.
asnTreeAddString	Adds a string object to the current active node of the ASN tree.
asnTreeCreate	Creates a tree in memory to hold an ASN.1 file structure.
asnTreeDelete	Deletes the last created or used ASN.1 tree.
asnTreeDeleteNodeByIndex	Deletes a node from the ASN.1 tree.
asnTreeFlush	Flushes the content of the ASN.1 tree to the output.
asnTreeGetStoredNode	Gets the active (working) node from a list of created and store
asnTreePop	Backs up one level in the ASN.1 tree hierarchy.

Table 85-2 (Cont.) ASN.1 Functions

Function	Description
asnTreePushTag	Adds a new block to the current active node of the ASN.1 tree and sets this new block as an active node of the tree.
asnTreeStoreNode	Stores an index to a constructed block node in the ASN.1 tree, when the block position in the tree is fixed.

asnTreeAddInteger

This function adds an integer object to the current active node of the ASN.1 tree.

Syntax

```
Bool asnTreeAddInteger(String blockName, Long value);
```

Parameters

blockName
The name of the block to add (exact type from the block description file).

value
The integer to insert as the value.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreeAddInteger("TAP3.DataVolume.DATA_VOLUME", 2512);
...
```

asnTreeAddString

This function adds a string object to the current active node of the ASN.1 tree.

Syntax

```
Bool asnTreeAddString(String blockName, String value)
```

Parameters

blockName
The name of the block to add. This must exactly match the type from the block description file.

value
The string to insert as the value.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreeAddString("TAP3.CalledPlace.CALLED_PLACE", "Freephone");
...
```

asnTreeCreate

This function creates a tree in memory to hold an ASN.1 file structure, where the Length field of the objects can be calculated in the end, just before writing on the output.

Parameters

None.

Return

True on success, otherwise, **False**

Only one tree can be in use at a time.

Example

```
...
if ( asnTreeCreate() == false )
{
logFormat( "asnTreeCreate() failed.");
}
...
```

asnTreeDelete

This function deletes the last created or used ASN.1 tree.

Syntax

```
Bool asnTreeDelete();
```

Parameters

None.

Return Values

Returns **True** if successful; otherwise returns **False**.

Example

```
...
if ( asnTreeDelete() == false )
{
logFormat( "asnTreeDelete() failed.");
}
...
```

asnTreeDeleteNodeByIndex

This function deletes a node from the ASN.1 tree, by recursively deleting all contained blocks and values.

Syntax

```
Bool asnTreeDeleteNodeByIndex(Long nodeIdx);
```

Parameter

nodeIdx
Node index in the ASN.1 tree as returned by asnTreeStoreNode().

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
//there is no need for this optional block (no data to store
//in it), so delete it
asnTreeDeleteNodeByIndex(networkInfoIdx);
...
```

asnTreeFlush

This function flushes the content of the ASN.1 tree to the output.

Syntax

```
Bool asnTreeFlush();
```

Parameters

None.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
if ( asnTreeFlush() == false )
{
logFormat( "asnTreeFlush() failed.");
}
...
```

asnTreeGetStoredNode

This function gets the active (working) node from a list of created and stored, but not filled, constructed blocks.

Syntax

```
Bool asnTreeGetStoredNode(Long nodeIdx);
```

Parameter

nodeIdx
Node index in the ASN.1 tree as returned by asnTreeStoreNode().

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreeGetStoredNode(networkInfoIdx);
//use asnTreeAddString() and asnTreeAddInteger() to update
//the TAP3.NetworkInfo block.
...
```

asnTreePop

This function backs up one level in the ASN.1 tree hierarchy. Every asnTreePushTag(xxxx) should have an associated asnTreePop(); it is like opening and closing brackets.

Syntax

```
Bool asnTreePop();
```

Parameters

None.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreePushTag("TAP3.AuditControlInfo");
...
asnTreePop(); //asnTreePushTag("TAP3.AuditControlInfo");
...
```

asnTreePushTag

This function adds a new block to the current active node of the ASN.1 tree and sets this new block as an active node of the tree. Use this function to create constructed ASN.1 objects, for example, **SEQUENCE** or **CHOICE**.

If the **isIndefiniteLength** parameter is set to true, the Length field of the ASN.1 object is set to 0x80 and 2 null bytes are appended to the Value field of the ASN.1 object.

Syntax

```
Bool asnTreePushTag(String blockName [, Bool isIndefiniteLength=false] );
```

Parameters

blockName
The name of the structured block to add (exact type from the block description file).

isIndefiniteLength
Flag to indicate that the generated block has to use indefinite lengths. The default is false, that is, it stores the exact size of the value field in the objects header.

Return Values

Returns **True** if successful; otherwise, returns **False**.

Example

```
...
asnTreePushTag("TAP3.AuditControlInfo");
...
```

asnTreeStoreNode

This function stores an index to a constructed block node in the ASN.1 tree, when for example, the data values that should be put in this block are unknown, but the block position in the tree is fixed.

Syntax

```
Long asnTreeStoreNode();
```

Parameters

None.

Return Values

Returns a node index that can be used with asnTreeGetStoredNode(nodeIdx) or asnTreeDeleteNodeByIndex(nodeIdx).

Example

```
...
asnTreePushTag("TAP3.NetworkInfo");
Long networkInfoIdx = asnTreeStoreNode();
//Nothing to do now, node will be updated after all //details are processed
asnTreePop(); //for asnTreePushTag("TAP3.NetworkInfo");
...
```

The following example iScript demonstrates how to create an output file in ASN.1 containing only a list of QoS requested objects (one per event data record (EDR)), with all field values set to **3**.

This is the content of the **OutGrammar.dsc** file. There should be an associated file describing the block structure that is here used, for example, **TAP3.QoSRequestedList**.

```
// The initial iScript code
iScript
{
  use EXT_AsnTree; // iScript extension to build a Tree of ASN.1 object
```

```
                    // used to fill the Length value of the ASN.1 bloc,
                    // before printing on output stream
}
// The definition of the grammar
Grammar
{
  edr_stream:
    header
    details
    trailer
  ;
  header:
    HEADER
    {
      asnTreeCreate();
      asnTreePushTag("TAP3.QoSRequestedList");
    }
  ;
  trailer:
    TRAILER
    {
      asnTreePop(); //for asnTreePushTag("TAP3.QoSRequestedList");
      asnTreeFlush();
      asnTreeDelete();
    }
  ;
  details:
    details
    DETAIL
    {
      asnTreePushTag("TAP3.QoSRequested");
      asnTreeAddInteger("TAP3.QoSDelay.QOS_DELAY", 3);
      asnTreeAddInteger("TAP3.QoSMeanThroughput.QOS_MEAN_THROUGHPUT", 3);
      asnTreeAddInteger("TAP3.QoSPeakThroughput.QOS_PEAK_THROUGHPUT", 3);
      asnTreeAddInteger("TAP3.QoSPrecedence.QOS_PRECEDENCE", 3);
      asnTreeAddInteger("TAP3.QoSReliability.QOS_RELIABILITY", 3);
      asnTreePop(); //for asnTreePushTag("TAP3.QoSRequested");
    }
  | /*EMPTY*/
  ;
}
```

Database Connection Functions

Table 85-3 contains database connection functions.

Table 85-3 Database Connection Functions

Function	Description
dbBeginTransaction	Starts a new transaction using the specified connection.
dbCloseConnection	Closes a connection to the Pipeline Manager database.
dbCloseResult	Closes a result handle after processing the result data.
dbCommitTransaction	Commits a transaction to a specific connection.
dbConnection	Establishes a connection to the Pipeline Manager database. The handle returned by this function should be used in future calls to the **dbExecute** function.
dbDataConnection	Connects the extension to a DBC_Database module.

Table 85-3 (Cont.) Database Connection Functions

Function	Description
dbError	Retrieves a description for the last error. This description is not reset after a valid call to one of the other database connection functions. Therefore, **dbError** should only be called directly after one of the other database connection functions fails.
dbExecute	Executes an SQL statement against the Pipeline Manager database.
dbNextResult	Switches the cursor to the next result for the result handle you specify.
dbNextRow	Switches the cursor to the next row in the current result.
dbRollbackTransaction	Rolls the current transaction back for the specified connection.

dbBeginTransaction

This function starts a new transaction using the specified connection.

Syntax

```
Bool dbBeginTransaction(Long conHandle);
```

Parameter

conHandle
The connection you want to use for the new transaction.

Return Values

Returns **true** if the transaction was successfully started. Returns **false** if the function fails.

Example

```
if ( dbBeginTransaction( conHandle ) == false )
{
  logFormat( "ERROR: failed to begin a new transaction: " \
  + dbError() );
}
```

dbCloseConnection

This function closes a connection to the Pipeline Manager database.

Syntax

```
Bool dbCloseConnection(Long conHandle);
```

Parameter

conHandle
The connection you want to close.

Return Values

Returns **true** if the connection was successfully closed. Returns **false** if the function fails.

Example

```
if ( dbCloseConnection( conHandle ) == false )
{
  logFormat( "ERROR: failed to close a connection: " + \
  dbError() );
}
```

dbCloseResult

This function closes a result handle after processing the result data.

Syntax

```
Bool dbCloseResult(Long resHandle);
```

Parameter

resHandle
The result handle you want to close.

Return Values

Returns **true** if the result handle was successfully closed. Returns **false** if the function fails.

Example

```
resHandle = dbExecute( "SELECT * FROM INT_SUBS_CLI" );
if ( resHandle == INVALID_RESULT )
{
  logFormat( "ERROR: dbExecute() failed: " + dbError() );
}
...

// Process the result data

...
dbCloseResult( resHandle );
```

dbCommitTransaction

This function commits a transaction to a specific connection.

Syntax

```
Bool dbCommitTransaction(Long conHandle);
```

Parameter

conHandle
The connection you want to use for the transaction.

Return Values

Returns **true** if the transaction was successfully committed to the connection. Returns **false** if the function fails.

Example

```
if ( dbCommitTransaction( conHandle ) == false )
{
  logFormat( "ERROR: failed to commit the transaction: " + dbError() );
}
```

dbConnection

This function establishes a connection to the Pipeline Manager database. The handle returned by this function should be used in future calls to the **dbExecute** function.

> **Note:**
>
> Before calling **dbConnection**, connect to DBC_Database module using **dbDataConnection**.

Syntax

```
Long dbConnection();
```

Parameters

None.

Return Values

Returns the handle for the new connection (the handle is a value greater than or equal to 0) if the function is successful. Returns **INVALID_CONNECTION** if the function fails.

Example

```
conHandle = dbConnection();
if ( conHandle == INVALID_CONNECTION )
{
  logFormat( "ERROR: dbConnection() failed: " + dbError() );
}
```

dbDataConnection

This function connects the extension to a DBC_Database module. This connection is valid for the whole extension; you cannot connect the extension to two different DBC_Database modules.

> **Note:**
>
> Before calling **dbConnection**, connect to DBC_Database module using **dbDataConnection**.

Syntax

```
Bool dbDataConnection(String dbcModule);
```

Parameter

dbcModule
The registry name for the DBC_Database module.

Return Values

Returns **true** if the extension was successfully connected to the module. Returns **false** if the function fails.

Example

```
use IXT_Db;

if ( dbDataConnection( "integrate.DataPool.Login.Module" ) == \
true )
{
  logFormat( "Connection to DBC module established" );
}
else
{
  logFormat( "ERROR: failed to establish the connection \
  to DBC module" );
}
```

dbError

This function retrieves a description for the last error. This description is not reset after a valid call to one of the other database connection functions. Therefore, **dbError** should only be called directly after one of the other database connection functions fails.

Syntax

```
String dbError();
```

Parameters

None.

Return Values

Returns a description of the error.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );
if ( resHandle == INVALID_RESULT )
{
  logFormat( "ERROR: dbExecute() failed: " + dbError() );
}
```

dbExecute

This function executes an SQL statement against the Pipeline Manager database. The handle this function returns should be used to access the result of the SQL statement in the **dbNextResult** and **dbNextRow** calls that follow. After processing the result data, free the handle by calling **dbCloseResult**.

Syntax

```
Long dbExecute(Long conHandle, String sqlStatement);
```

Parameters

conHandle
The connection you want to use.

sqlStatement
The SQL statement to execute.

Return Values

Returns the result handle (the handle is a value greater than or equal to 0) if the function is successful. Returns **INVALID_RESULT** if the function fails.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );
if ( resHandle == INVALID_RESULT )
{
  logFormat( "ERROR: dbExecute() failed: " + dbError() );
}
```

dbNextResult

This function switches the cursor to the next result for the result handle you specify.

> **Note:**
>
> This function is specific to results, not rows. The return generated by **dbExecute** can consist of a list of results in table form, with each result containing one or more data rows. Using **dbNextResult** moves the cursor from result to result, not from data row to data row within a result.

Syntax

```
Long dbNextResult(Long resHandle);
```

Parameter

resHandle
The result handle you want to process.

Return Values

Returns the next result in the result handle if the function is successful. Returns **NO_MORE_RESULTS** if the function reaches the last result. Returns a value less than 0 if the function fails.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );
```

```
// loop for all results
do
{
  // process the rows of the current result
  while ( (ret = dbNextResult( resHandle )) == NEXT_RESULT );

if ( ret != NO_MORE_RESULTS )
{
  logFormat( "ERROR: dbNextResult() failed: " + dbError() );
}
```

dbNextRow

This function switches the cursor to the next row in the current result.

> **Note:**
>
> This function is specific to rows, not results. The return generated by **dbExecute** can consist of a list of results in table form, with each result containing one or more data rows. Using **dbNextRow** moves the cursor from row to row within a result, not from result to result.

Syntax

```
Long dbNextRow(Long resHandle, ...);
```

Parameters

resHandle
The handle for the result you want to process.
A list of bound variables

Return Values

Returns the next row in the result if the function is successful. Returns **NO_MORE_ROWS** if the function reaches the last row. Returns a value less than 0 if the function fails.

Example

```
resHandle = dbExecute( conHandle, "SELECT * FROM INT_SUBS_CLI" );

// loop for all rows
while ( (rowRet = dbNextRow( resHandle, cli, validFrom validTo )) > 0 )
{
  ...
}

if ( rowRet != NO_MORE_ROWS )
{
  logFormat( "ERROR: dbNextRow() failed: " + dbError() );
}
```

dbRollbackTransaction

This function rolls the current transaction back for the specified connection.

Syntax

```
Bool dbRollbackTransaction(Long conHandle);
```

Parameter

conHandle
The connection whose transaction you want rolled back.

Return Values

Returns **true** if the rollback is successful. Returns **false** if the function fails.

Example

```
if ( dbRollbackTransaction( conHandle ) == false )
{
  logFormat( "ERROR: failed to rollback current transaction: " \
  + dbError() );
}
```

Data Normalizing Functions

Table 85-4 contains data normalizing functions.

Table 85-4 Data Normalizing Functions

Function	Description
convertCli	Normalizes wireline and wireless command-line interfaces (CLIs). Static class function: "EXT_ConvertCli::convert "
convertIPv4	Normalizes IPv4 addresses. Static class function: "EXT_ConvertIPv4::convert "
String convertIPv6	Normalizes IPv6 addresses. Static class function: "EXT_ConvertIPv6::convert "
convertIPv4onv6	Normalizes IPv4 over IPv6 addresses. Static class function: "EXT_ConvertIPv4onv6::convert "

convertCli

This function normalizes wireless and wireline CLIs into international format.

Syntax

```
String convertCli( String cli,
                   String modInd,
                   Long typeOfNumber,
                   String natAccessCode,
                   StringArray intAccessCode,
                   StringArray countryCode,
                   String intAccessCodeSign,
                   String natDestinCode )
```

Parameters

cli
CLI to normalize.

modInd
Modification Indicator, for example, "00".

typeOfNumber
Type Of Number, for example, 0.

natAccessCode
National Access Code, for example, "0".

intAccessCode
International Access Code, for example, "00".

countryCode
Country Code, for example, "49".

intAccessCodeSign
International Access Code Sign, for example, "+".

natDestinCode
National Destination Code, for example, "172".

Return Values

Returns a CLI in international normalized format: <iac>< cc><ndc>extension.

Example

```
...
use EXT_Converter;

String normCli;
String cli = "01721234567";

normCli = convertCli( cli, "00", 0, "0", "00", "49", "+", "172" );

// normCli now contains: 00491721234567

...
```

convertIPv4

This function normalizes IPv4 addresses.

Syntax

```
String convertIPv4( String ip );
```

Parameter

ip
The IP address to normalize.

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 3 digits with zeroes.

Example

```
....
use EXT_Converter;

String normIp;
String ip = "192.168.1.253";

normIp = convertIPv4( ip );

// normIp now contains: 192168001253

...
```

String convertIPv6

This function normalizes IPv6 addresses.

Syntax

```
String convertIPv6(String ip;
```

Parameter

ip
The IP address to normalize

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
use EXT_Converter;

String normIp;
String ip = "0:0:0:AF:E:0:1:FE";

normIp = convertIPv6( ip );

// normIp now contains: 00000000000000AF000E0000000100FE

...
```

convertIPv4onv6

This function normalizes IPv4 over IPv6 addresses. The decimal IPv4 address is converted into hexadecimal representation.

Syntax

```
String convertIPv4onv6(String ip);
```

Parameter

ip
The IP address to normalize

Return Values

Returns an IPv6 address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
use EXT_Converter;

String normIp;
String ip = "0:0:0:0:0:0:192.168.10.1";

normIp = convertIPv4onv6( ip );

// normIp now contains: 00000000000000000000000C0A80A01

...
```

Date Functions

Table 85-5 contains date functions.

Table 85-5 Date Functions

Function	Description
dateAdd	Adds date and time values.
dateDiff	Calculates the difference between two dates.
dateIsValid	Checks a date for validity; for example, after initialization from a string.
dateToStr	Converts a date value to a string.
strToDate	Converts a string into a date value.
sysdate	Retrieves the current system date.

dateAdd

This function manipulates date and time values.

Syntax

```
Date dateAdd(Date source [, Long years [, Long months [, days [, Long hours [, Long
mins [, Long secs]]]]]]);
```

Parameters

source
The source date for the addition.

years
The number of years to add. This parameter can be positive or negative.

months
The number of months to add. This parameter can be positive or negative.

days
The number of days to add. This parameter can be positive or negative.

hours
The number of hours to add. This parameter can be positive or negative.

mins
The number of minutes to add. This parameter can be positive or negative.

secs
The number of seconds to add. This parameter can be positive or negative.

Return Values

Returns the manipulated source date.

> **Note:**
>
> The variable source itself is not manipulated; only the result is returned.

Example

```
Date now = sysdate();
Date later = dateAdd( now, 1, 2, 0, 5 );

logStdout( "Date now is " + dateToStr(now) + "\n" );
logStdout( "In 1 year, 2 months and 5 hours it is " + dateToStr(later) + "\n" );
```

dateDiff

This function calculates the difference between two dates. The difference is returned in seconds.

Syntax

```
Long dateDiff(Date date1, Date date2);
```

Parameters

date1
The first date used for calculating the difference. This is the minuend.

date2
The second date used for calculating the difference. This is the subtrahend.

Return Values

Returns the difference between the first and second date, in seconds.

Example

```
if ( dateDiff( sysdate(), date ) < 0 )
{
  logFormat( "the date is a future date" );
}
```

dateIsValid

This function checks a date for validity; for example, after initialization from a string.

Syntax

```
Bool dateIsValid(Date date);
```

Parameter

date
The date to validate.

Return Values

Returns **true** if the date is valid. Returns **false** if the date is not valid.

Example

```
Date timeStamp = strToDate( timeString );
if ( dateIsValid( timeStamp ) == false )
{
  logFormat( timeString + " is no valid date string" );
}
```

dateToStr

This function converts a date value to a string.

Syntax

```
String dateToStr(Date date);
```

Parameters

%a
The abbreviated week day name; for example, Sun for Sunday. This is from tm::tm_wday.

%A
The full weekday name; for example, Sunday. This is from tm::tm_wday.

%b
The abbreviated month name; for example, Feb for February.

%B
The full month name; for example, February.

%c
The date and time; for example, Feb 29 14:34:56 2004. This may use all members.

%d
The day of the month; for example, 29.

%H
The hour of the 24-hour day; for example, 14.

%I
The hour of the 12-hour day; for example, 02.

%j
The day of the year starting from 001; for example, 060. This is from tm::tm_yday.

%m
The month of the year, from 01; for example, 02.

%M
The minutes after the hour; for example, 34.

%p
The AM/PM indicator, if any; for example, AM.

%S
The seconds after the minute; for example, 56.

%U
The week of the year, starting from 00; for example, 45. This is from tm::tm_yday and tm::tm_wday. The week is defined as starting on Sunday.

%w
The day of the week, with 0 for Sunday; for example, 2 for Tuesday.

%W
The week of the year, from 00; for example, 33. This is from tm::tm_yday and tm::tm_wday. In this case, the week is defined as starting on Monday.

%x
The date; for example, Feb 29 2004. This uses tm::tm_yday in some locales.

%X
The time; for example, 14:34:56.

%y
The year of the century, from 00; for example, 04 for 2004. In most cases, you should avoid this parameter; to ensure correct handling of the past century, use **%Y** instead.

%Y
The year including the century; for example, 1994.

%Z
The time zone name; for example, PST or PDT. This is from tm::tm_isdst.

Return Values

Returns the date as a string using the format defined by the function parameters if the function is successful. Returns an empty string if the date is invalid.

Example

```
dateToString("%a %d. %B %Y")
```

will result in:

```
"Mon 24. June 2002"
```

strToDate

This function converts a string into a date value. The only supported string format is *YYYYMMDDHHMMSS*.

Syntax

```
Date strToDate(String dateStr);
```

Parameters

%%
The literal **%** character.

%d
The day of the month; for example, 29. The range is 00-31.

%H
The hour of the 24-hour day; for example, 14. The range is 00-23.

%m
The month of the year, from 01; for example, 02. The range is 01-12.

%M
The minutes after the hour; for example, 34. The range is 00-59.

%S
The seconds after the minute; for example, 56. The range is 00-59.

%y
The year of the century, from 00; for example, 04 for 2004. The range is 01-99. In most cases, you should avoid this parameter.

%Y
The year including the century; for example, 1994.

Return Values

Returns a valid date if the input string is in the right format. Returns an invalid date if the format is not correct.

Example

```
edrDate(DETAIL.CHARGING_START_TIMESTAMP) = \
strToDate("24.12.2002", "%d. %m. %Y");
```

sysdate

This function retrieves the current system date.

Syntax

```
Date sysdate();
```

Parameters

None.

Return Values

Returns the current system date.

Example

```
Date now;
now = sysdate();
```

EDR Container Functions

Table 85-6 contains EDR container functions.

Table 85-6 EDR Container Functions

Function	Description
edrAddAdditionalStream	Adds additional output streams to each EDR.
edrAddDatablock	Adds a new data block to the current EDR container.
edrAddDatablockEx	Adds a new data block to the current EDR container.
edrAddError	Adds a new error to the current EDR container.
edrArrayIndex	Accesses the array index values in EDR container.
edrClearErrors	Clears the list of errors that the pipeline modules add to the EDR container.
edrConnectToken	Associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field. This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.
edrConnectTokenEx	Associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field. This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.
edrContainsAdditionalStream	Determines whether an EDR has an additional output stream with the name you pass in.
edrCurrentTokenIndex	Provides the index of the token parsed from the stream. Valid only in input grammar.
edrDate	Retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values.

Table 85-6 (Cont.) EDR Container Functions

Function	Description
edrDateEx	Retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values.
edrDecimal	Retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values.
edrDecimalEx	Retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values.
edrDelete	Deletes the current EDR container, changing the current pointer to the EDR container directly in front of the deleted EDR.
edrDeleteDatablock	Deletes a data block from the current EDR container. The function is not supported for nested transactions.
edrDeleteField	Clears the contents of a field in an EDR container. The function is not supported for nested transactions.
edrDuplicate	Duplicates the current EDR container.
edrEmptyInput	Sets the state of a field to **EDR_INPUT_EMPTY** when the field is present in the CDR but contains no value.
edrFieldConnectInfo	Retrieves the Info string associated with the token for the corresponding EDR field. By default, the Info string contains the description of the token type. The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.
edrFieldTokenBytePos	Calculates the position of the token associated with the corresponding EDR field. The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.
edrGetAdditionalStream	Gets the name of an additional EDR output stream given an array index number.
edrGetError	Retrieves the names of the attached error messages.
edrGetErrorParameters	Retrieves the parameters associated to a specified error.
edrGetErrorSeverity	Retrieves the severity for each of the associated errors.
edrGetStream	Gets the output stream for an EDR.
edrHasError	Retrieves the names of the attached error messages.
edrInputState	Retrieves the input state of an EDR field.
edrInternalState	Returns the internal state of an EDR field.
edrInternalStateEx	Returns the internal state of an EDR field.
edrIsValidDetail	Determines whether the current EDR container is a valid detail container.
edrLong	Retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values.
edrLongEx	Retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values.
edrMaxSeverity	Finds the maximum severity of the errors added to the current EDR container.

Table 85-6 (Cont.) EDR Container Functions

Function	Description
edrMissingInput	Sets the state of a field to **EDR_INPUT_MISSING** when the field is not present in the CDR.
edrNumDatablocks	Determines the number of data blocks of the specified type.
edrNumDatablocksEx	Determines the number of data blocks of the specified type.
edrNumErrors	Accesses the number of error messages attached to the current EDR container.
edrNumTokens	Accesses the number of tokens attached to the current EDR container.
edrRemoveAdditionalStream	Removes additional output streams from an EDR.
edrSetContentType	Sets the content type of the current EDR container.
edrSetCurrent	Sets the current EDR container.
edrSetIsValidDetail	Sets the EDR container's valid detail flag. The valid detail flag specifies whether the EDR container is to be discarded.
edrSetStream	Sets the output stream for an EDR.
edrString	Retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values.
edrStringEx	Retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values.
edrTokenString	Used to retrieve the content of each token, as identified by their indexes. When the index is not available, as for a function call with no argument, this function returns the complete byte string attached to the EDR. The byte string corresponds to the original input string that generated the EDR. The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.
iRulesModeOn	Enables the iRules mode.
iRulesModeOff	Disables the iRules mode.
pipelineName	Retrieves the name of the pipeline in which the script is running.
stopPipeline	Stops the pipeline from which it is called.

edrAddAdditionalStream

This function adds additional output streams to each EDR.

Each Out_GenericStream pipeline module has a default output stream for EDRs. You use this function to add additional output streams to direct the output to additional locations.

Output stream characteristics (output path, record prefix, and record suffix) are set in the registry file.

If the stream name sent in with this function already exists, **edrAddAdditionalStream** returns **true** but does not create the stream again.

Syntax

```
Bool edrAddAdditionalStream(String output_stream_name);
```

Parameter

output_stream_name
The name of the new output stream that you are adding.

Return Values

Returns **true** if the function is successful. Returns **false** for all other conditions.

Example

This iScript example adds two additional output module streams:

```
addoutmod.isc
------
function onDetailEdr
{
      if (edrAddAdditionalStream( "TElOut1" ) == true)
   {
      logStdout("Stream TelOut1 added ");
   }
      if (edrAddAdditionalStream( "TELOut2" ) == true)
   {
      logStdout("Stream TElOut2 added ");
   }
} // end onDetailEdr + end iScript ----
```

This registry fragment shows the two example iScript files, **addoutmod.isc** and **removeoutmod.isc**, defined in the FunctionPool section. These iScripts add and remove output module streams. The new iScripts are shown in **bold**.

```
   FunctionPool
   {
      Iscript
      {
         ModuleName = FCT_IScript
         Module
         {
            Active = True
            Source = FILE
            Scripts
            {
               addoutmod
               {
                  FileName = ./samples/simple/addoutmod.isc
               }
            }
         }
      }
      Iscript2
      {
         ModuleName = FCT_IScript
         Module
         {
            Active = True
            Source = FILE
            Scripts
            {
               removeoutmod
               {
```

```
                FileName = ./samples/simple/removeoutmod.isc
            }
        }
    }
}
```

This output registry section defines the **TELOut1** output section:

```
TELOut1
{
   ModuleName = OUT_GenericStream
   Module
   {
      Grammar = ./formatDesc/Formats/Solution42/SOL42_V430_OutGrammar.dsc
      DeleteEmptyStream = TRUE
      OutputStream
      {
         ModuleName = EXT_OutFileManager
         Module
         {
            OutputPath = ./samples/simple/data/out2
            OutputPrefix = Sol42_
            OutputSuffix = .out

            TempPrefix = tmp
            TempDataPath = ./samples/simple/data/out2
            TempDataPrefix = out.tmp.
            TempDataSuffix = .data

            Replace = TRUE
         }
      }
   }
}
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputPrefix, and OutputSuffix values for each output stream defined in the registry.

edrAddDatablock

This function adds a new data block to the current EDR container.

Syntax

```
Bool edrAddDatablock(EdrField block [, Long idx1 [, Long idx2 ...]]);
```

Parameters

block
The name of the EDR block you want to add.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
if ( edrAddDatablock( DETAIL.ASS_CBD ) == false )
{
  logFormat( "ERROR: failed to add ASSOCIATED_CHARGE \
  datablock" );
}
```

edrAddDatablockEx

This function adds a new data block to the current EDR container.

Syntax

```
Bool edrAddDatablockEx(String block, Long indicesArray, Long numIndices);
```

Parameters

block
The name of the EDR block you want to add.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.ASS_CBD";
numberOfIndices = 0;

if ( edrAddDatablockEx(edrFieldName, indicesArray, numberOfIndices) == false )
{
  logFormat( "ERROR: failed to add ASSOCIATED_CHARGE \
  datablock" );
}
```

edrAddError

This function adds a new error to the current EDR container.

Syntax

```
Bool edrAddError(String error, Long severity [, String paramX...]);
```

Parameters

error
The name of the error you want to add to the EDR container.

severity
The severity of the error:

- **0** = Debug
- **1** = Normal
- **2** = Warning
- **3** = Minor error
- **4** = Major error
- **5** = Critical error

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
if ( edrString( DETAIL.SERVICE_CODE ) != "Tel" and \
edrString( DETAIL.SERVICE_CODE ) != "Fax" )
{
  edrAddError( "ERR_UNKNOWN_SERVICE_CODE", 3, edrString\
  ( DETAIL.SERVICE_CODE ) );
}
```

edrArrayIndex

This function accesses array index values in EDR container.

Syntax

```
Long edrArrayIndex(EdrField block [, Long idx1 [, Long idx2 ...]]);
```

Parameters

block
The array block of the EDR container whose index you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the index of the EDR container.

Example

```
edrArrayIndex( DETAIL.ASS_TCF_AAA_DETAIL.PCM_OP_TCF_AAA_AUTHORIZE.INPUT.PIN_FLD_BALANCES,
 0, 0, 0, 0) = 1;
edrIndex =
edrArrayIndex( DETAIL.ASS_TCF_AAA_DETAIL.PCM_OP_TCF_AAA_AUTHORIZE.OUTPUT.PIN_FLD_BALANCES
, 0, 0, 0, 0);
```

edrClearErrors

This function clears the list of errors that the pipeline modules add to the EDR container.

Each pipeline module error has a name, severity level, and optional parameters that you can use for debugging or constructing an error message. The error list is a collection of all the errors that the pipeline modules have added to an EDR, the number of errors in the list, and the maximum severity of the errors. You can use the errors to reject an EDR or to instruct the pipeline module to process an EDR differently or to not process an EDR.

However, if an EDR does not have errors severe enough to be rejected or processed differently, you can use this function to remove the errors from the list. This function resets the error count to 0 and the maximum severity level to normal.

> **Note:**
>
> Before clearing the errors, analyze all the errors in the EDR to ensure they can be safely ignored.

Syntax

```
Void edrClearErrors();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
function onInvalidDetailEdr
{
  if(edrNumErrors() > 0)
   {
   logStdout(" Current Edr contains" + longToStr(edrNumErrors()) +      "Errors");
   edrClearErrors();
   logStdout(" Current Edr contains" + longToStr(edrNumErrors()) +       "Errors after
clearErrors");
   }
  else
   {
   logStdout(" Current Edr contains no Errors");
   }
}
```

edrConnectToken

This function associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field.

This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.

Syntax

```
Bool edrConnectToken(EdrField field [, Long idx1 [, Long idx2 ...]], const String tokenName);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

tokenName
The name of the token field to access (stream record field).

Return Values

Returns **true** if the EDR field is successfully associated with the input token. Returns **false** if the function fails.

Example

```
Bool success = edrConnectToken(DETAIL.RECORD_TYPE, "SOL42.DETAIL.RECORD_NUMBER");
```

edrConnectTokenEx

This function associates an EDR field with an input token and is identical to calling a block mapping with **edrInputMap**, except that it is accomplished using only one field.

This function calls the **edrMissingInput** and **edrEmptyInput** state-setting functions, which indicate the reason for missing fields.

Syntax

```
Bool edrConnectTokenEx(String field, Long indicesArray, Long numIndices, String tokenName);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

tokenName
The name of the token field to access (stream record field).

Return Values

Returns **true** if the EDR field is successfully associated with the input token. Returns **false** if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.RECORD_TYPE";
numberOfIndices = 0;

Bool success = edrConnectTokenEx(edrFieldName, indicesArray, numberOfIndices,
"SOL42.DETAIL.RECORD_NUMBER");
```

edrContainsAdditionalStream

This function determines whether an EDR has an additional output stream with the name you pass in. EDRs contain one default stream and any number of additional output streams.

Syntax

```
Bool edrContainsAdditionalStream(String output_stream_name);
```

Parameter

output_stream_name
The name of the output stream you want to confirm exists in the EDR.

Return Values

Returns **true** if the stream exists. Returns **false** if it does not.

Example

```
if ( edrContainsAdditionalStream( "TELOut3" ) == false )
{
logStdout( "ERROR: EDR does not contain additonal stream: TELOut1\n" );
}
```

edrCurrentTokenIndex

This function returns the index of the token parsed from the stream. It is valid only in input grammar.

Syntax

```
Long edrCurrentTokenIndex();
```

Parameters

None.

Return Values

Returns the token index if the token exists. Returns **-1** if the function fails.

Example

```
Long index = edrCurrentTokenIndex();
logStdout("Currently processing: " + edrTokenString(index0 + "\n");
```

edrDate

This function retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values. When setting date values, use the function as the left-hand value in an assignment statement.

Syntax

```
Date edrDate(EdrField field [, Long idx1 [, Long idx2 ... ]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the date value of the EDR field if the function is successful. Returns **INVALID_DATE** if the data type for this EDR is not **Date** or if the path through the EDR tree structure is not valid.

Example

```
Date timeStamp;

timeStamp = edrDate( DETAIL.CHARGING_START_TIMESTAMP ); \
edrDate( DETAIL.CHARGING_START_TIMESTAMP ) = sysdate();
```

edrDateEx

This function retrieves and sets date values in the current EDR container. This function is usually used to retrieve date values. When setting date values, use the function as the left-hand value in an assignment statement.

Syntax

```
Date edrDateEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices

Return Values

Returns the date value of the EDR field if the function is successful. Returns **INVALID_DATE** if the data type for this EDR is not **Date** or if the path through the EDR tree structure is not valid.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.CHARGING_START_TIMESTAMP";
numberOfIndices = 0;

Date timeStamp;

timeStamp = edrDateEx( edrFieldName, indicesArray, numberOfIndices); \
edrDateEx( edrField, indicesArray, numberOfIndices) = sysdate();
```

edrDecimal

This function retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values. When used to set decimal values, use the function as the left-hand value in an assignment statement.

Syntax

```
Decimal edrDecimal(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the decimal value of the EDR field if the function is successful. Returns an invalid decimal value if the data type for this EDR is not decimal or if the path through the EDR tree structure is not valid (for example, an index number is wrong).

Example

```
Decimal oldAmount;

oldAmount = edrDecimal( DETAIL.CHARGED_AMOUNT_VALUE ); \
edrDecimal( DETAIL.CHARGED_AMOUNT_VALUE ) = oldAmount + 1.0;
```

edrDecimalEx

This function retrieves and sets decimal values in the current EDR container. This function is used usually to retrieve decimal values. When used to set decimal values, use the function as the left-hand value in an assignment statement.

Syntax

```
Decimal edrDecimalEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

level
Number of indices.

Return Values

Returns the decimal value of the EDR field if the function is successful. Returns an invalid decimal value if the data type for this EDR is not decimal or if the path through the EDR tree structure is not valid (for example, an index number is wrong).

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.CHARGED_AMOUNT_VALUE";
numberOfIndices = 0;

Decimal oldAmount;

oldAmount = edrDecimalEx(edrFieldName, indicesArray, numberOfIndices); \
edrDecimalEx(edrFieldName, indicesArray, numberOfIndices) = oldAmount + 1.0;
```

edrDelete

This function deletes the current EDR container, changing the current pointer to the EDR container directly in front of the deleted EDR.

Syntax

```
Bool edrDelete();
```

Parameters

None.

Return Values

Returns **true** if the current EDR container is deleted successfully. Returns **false** if there was no current EDR container.

Example

```
if ( edrDelete() )
{
  logStdout( "EDR container deleted" );
}
```

edrDeleteDatablock

This function deletes a data block from the current EDR container. The function is not supported for nested transactions (for example, transactions contained within transactions).

Syntax

```
Bool edrDeleteDatablock(EdrField block, Long idx1 [, Long idx2 ...]);
```

Parameters

block
The name of the data block you want to delete.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the data block is successfully deleted. Returns **false** if the operation fails.

Example

```
if edrDeleteDatablock( DETAIL.ASS_GSMW_EXT, 0 ) == false )
{
  logStdout("Error: failed to delete datablock");
}
```

edrDeleteField

This function clears the contents of a field in an EDR container. The function is not supported for nested transactions (for example, transactions contained within transactions).

Syntax

```
Bool edrDeleteField(EdrField field, Long idx1 [, Long idx2 ...]);
```

Parameters

field
The name of the EDR field you want to delete.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the EDR field content is successfully deleted. Returns **false** if the operation fails.

Example

```
if edrDeleteField( DETAIL.ASS_GSMW_EXT.RECORD_NUMBER ) == false )
{
  logStdout("ERROR: failed to delete field");
}
```

edrDuplicate

This function duplicates the current EDR container. The returned index is used as a parameter for the **edrSetCurrent** function to access the newly created EDR container.

Syntax

```
Long edrDuplicate();
```

Parameters

None.

Return Values

Returns the index of the duplicate EDR container (the index is greater than or equal to 0) if the function is successful. Returns a value less than 0 if the function fails.

Example

```
Long index = edrDuplicate();
if ( index < 0 )
{
  logFormat( "ERROR: duplication of edr failed" );
}
else
{
  if ( edrSetCurrent( index ) == true )
  {
    // send new edr to duplicate output
    edrSetStream( "DuplicateOutput" );
  }
}
```

edrEmptyInput

This function sets the state of a field to **EDR_INPUT_EMPTY** when the field is present in the CDR but contains no value.

Syntax

```
Bool edrEmptyInput(EdrField field, Long idx1 [, Long idx2 ... ]);
```

Parameters

field
The name of the empty EDR field.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
Bool success = edrEmptyInput(DETAIL.BASIC_SERVICE);
```

edrFieldConnectInfo

This function retrieves the Info string associated with the token for the corresponding EDR field. By default, the Info string contains the description of the token type. This is the default for ASCII object types.

The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.

Syntax

```
String edrFieldConnectInfo(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the Info string associated with the token for the EDR field if the function is successful. Returns an empty string if the path through the EDR tree structure is not valid.

Example

```
logStdout("This field is of type: "+ edrFieldConnectInfo\
( DETAIL.RECORD_TYPE ) +"\n" );
```

edrFieldTokenBytePos

This function calculates the position of the token associated with the corresponding EDR field. The calculation is in bytes starting from the beginning of the input file.

The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.

Syntax

```
Long edrFieldTokenBytePos(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the position (in bytes) of the token associated with the EDR field if the function is successful. Returns **-1** if the EDR field is not associated with a token.

Example

```
if ( edrString( DETAIL.RECORD_TYPE ) != "020" )
{
  logStdout("Error, unexpected value at bytePosition= \
  "+ longToStr(edrFieldTokenBytePos( DETAIL.RECORD_TYPE )) + \
  "\n" );
}
```

edrGetAdditionalStream

This function gets the name of an additional EDR output stream given an array index number.

Each EDR contains a default output stream and any number of additional output streams.

Syntax

```
String edrGetAdditionalStream(Long index_number);
```

Parameter

index_number
The array index of the output stream that you need the name of.

Return Values

Returns the name of the stream if the function is successful. Returns an empty string for all other conditions.

Example

```
String streamName = edrGetAdditionalStream( 5)

if ( streamName == "" )
{
logStdout( "ERROR: no additional stream set at index: 5\n" );
}
```

edrGetError

This function retrieves the names of the attached error messages.

Syntax

```
String edrGetError(Long idx);
```

Parameter

idx
The index of the error to be retrieved.

Return Values

Returns the name of the attached error if the function is successful. Returns an empty string if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + ": " + \
  edrGetError(i) + "\n" );
}
```

edrGetErrorParameters

This function retrieves the parameters associated to a specified error.

Syntax

```
Long edrGetErrorParameters(Long idx, Array params);
```

Parameters

idx
The index of the error that you want to retrieve, where 0 <= *idx* < **edrNumErrors**.

params
The string array where the parameters can be stored. This is a return parameter.

Return Values

Returns the number of parameters in the array. Returns **0** if this function fails or if there are no parameters in the array.

Example

```
String paramList[];
Long paramCount;
Long Tap3MaxParamCount = 7;
long i;
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  if (edrGetError(i) == "ERR_TAP3_RET")
  {
    // get parameter list
    paramCount = edrGetErrorParameters(i, paramList);
    // check if enough parameters
    if (paramCount != Tap3MaxParamCount)
    {
      logStdout( "ERROR " + longToStr(i) + ": " + edrGetError(i) \
      + ", has missing parameters\n" );
    }
  }
}
```

edrGetErrorSeverity

This function retrieves the severity for each of the associated errors.

Syntax

```
Long edrGetErrorSeverity(Long idx);
```

Parameter

idx
The index of the error whose severity is being retrieved.

Return Values

Returns **0** if the severity of the attached error is Normal. Returns **1** if the severity of the attached error is Warning. Returns **2** if the severity of the attached error is Minor. Returns **3** if the severity of the attached error is Major. Returns **4** if the severity of the attached error is Critical. Returns **-1** if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + " Severity: " + \
  longToStr(edrGetErrorSeverity(i)) + "\n" );
}
```

edrGetStream

This function gets the output stream for an EDR.

Syntax

```
String edrGetStream();
```

Parameters

None.

Return Values

Returns the name of the actual string.

Example

```
String streamName = edrGetStream();
```

edrHasError

This function retrieves the names of the attached error messages.

Syntax

```
Bool edrHasError(String error);
```

Parameter

error
The name of the error to be retrieved.

Return Values

Returns the name of the attached error if the function is successful. Returns an empty string if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + ": " + \
  edrGetError(i) + "\n" );
}
```

edrInputState

This function retrieves the input state of an EDR field.

Syntax

```
Long edrInputState(EdrField field, Long idx1 [, Long idx2...]);
```

Parameters

field
The name of the EDR field for which to return the input state.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **1** if the EDR field contains a default value that was added due to missing input data in the CDR. Returns **2** if the EDR field contains a default value that was added due to empty input data in the CDR. Returns **3** if the EDR field is not populated or contains data that came from the CDR.

Example

```
Bool boolvar;
boolvar = edrEmptyInput(DETAIL.BASIC_SERVICE);
boolvar = edrMissingInput(DETAIL.QOS_USED);
switch(edrInputState(DETAIL.BASIC_SERVICE))
{
  case EDR_INPUT_MISSING:
    logStdout("DETAIL.BASIC_SERVICE: MISSING\n");
    break;
  case EDR_INPUT_EMPTY:
    logStdout("DETAIL.BASIC_SERVICE: EMPTY\n");
    break;
  default:  // "uninteresting" values
    logStdout("DETAIL.BASIC_SERVICE: OTHER\n");
    break;
}
```

edrInternalState

This function returns the internal state of an EDR field.

Syntax

```
Long edrInternalState(EdrField field, Long idx1 [, Long idx2...]);
```

Parameters

field
The name of the EDR field for which to return the internal state.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns 0 if cleared. Returns 1 if connected. Returns 2 if initialized. Returns 3 if set. Returns 4 if restored. Returns 5 if restored asset. Returns -1 if the function fails.

Example

```
Long state = edrInternalState(DETAIL.ASS_CBD.CP.CHARGE);
```

edrInternalStateEx

This function returns the internal state of an EDR field.

Syntax

```
Long edrInternalStateEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns 0 if cleared. Returns 1 if connected. Returns 2 if initialized. Returns 3 if set. Returns 4 if restored. Returns 5 if restored asset. Returns -1 if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.ASS_CBD.CP.CHARGE";
indicesArray[0]=0;
indicesArray[1]=0;
numberOfIndices=2;

Long state = edrInternalStateEx(edrFieldName, indicesArray, numberOfIndices);
```

edrIsValidDetail

This function determines whether the current EDR container is a valid detail container. This helps you avoid processing of EDR containers that will be discarded.

Syntax

```
Bool edrIsValidDetail();
```

Parameter

None.

Return Values

Returns **true** if the current EDR container is a valid detail container. Returns **false** if it is not a valid detail container.

Example

```
if ( edrIsValidDetail() == true )
{
  // process the edr
}
```

edrLong

This function retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values. When setting Long values, use the function as left-hand value in an assignment statement.

Syntax

```
Long edrLong(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the Long value of the EDR field if the function is successful. Returns **0** if the EDR has no Long field or if the path through the EDR tree structure is not valid.

Example

```
edrLong( DETAIL.CHARGED_TAX_RATE ) = 1600;
```

edrLongEx

This function retrieves and sets Long values in the current EDR container. This function is usually used to retrieve Long values. When setting Long values, use the function as left-hand value in an assignment statement.

Syntax

```
Long edrLongEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the Long value of the EDR field if the function is successful. Returns **0** if the EDR has no Long field or if the path through the EDR tree structure is not valid.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.CHARGED_TAX_RATE";
numberOfIndices=0;
edrLongEx(edrFieldName, indicesArray, numberOfIndices) = 1600;
```

edrMaxSeverity

This function finds the maximum severity of the errors added to the current EDR container.

Syntax

```
Long edrMaxSeverity();
```

Parameters

None.

Return Values

Returns the maximum severity of the errors of the EDR container if the function is successful. Returns **0** if there are no errors. Returns **-1** if the function fails.

Example

```
if ( edrMaxSeverity() == 0 )
{
  // The edr has no errors with severity > 0
}
```

edrMissingInput

This function sets the state of a field to **EDR_INPUT_MISSING** when the field is not present in the CDR.

Syntax

```
Bool edrMissingInput(EdrField field, Long idx1 [, Long idx2...]);
```

Parameters

field
The name of the missing EDR field.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
Bool success = edrMissingInput(DETAIL.QOS_USED);
```

edrNumDatablocks

This function determines the number of data blocks of the specified type.

Syntax

```
Long edrNumDatablocks(EdrField block [, Long idx1 [, Long idx2 ...]]);
```

Parameters

block
The name of the data block you want to access.

idxN
Additional index values specifying the path through the EDR tree structure.

Return Values

Returns the number of data blocks (the number is greater than or equal to 0) if the function is successful. Returns a value less than 0 if the function fails.

Example

```
for ( i = 0; i < edrNumDatablocks( DETAIL.ASS_CBD ); i = i + 1 )
{
  String recordType = edrString( DETAIL.ASS_CBD.RECORD_TYPE, i );
}
```

edrNumDatablocksEx

This function determines the number of data blocks of the specified type.

Syntax

```
Long edrNumDatablocksEx(String block, Long indicesArray, Long numIndices);
```

Parameters

block
The name of the data block you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the number of data blocks (the number is greater than or equal to 0) if the function is successful. Returns a value less than 0 if the function fails.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.ASS_CBD";
numberOfIndices=0;

for ( i = 0; i < edrNumDatablocksEx(edrFieldName, indicesArray, numberOfIndices); i = i
+ 1 )
{
  String recordType = edrString( DETAIL.ASS_CBD.RECORD_TYPE, i );
}
```

edrNumErrors

This function accesses the number of error messages attached to the current EDR container.

Syntax

```
Long edrNumErrors();
```

Parameters

None.

Return Values

Returns the number of attached error messages (this number will be greater than or equal to 0) if the function is successful. Returns **-1** if the function fails.

Example

```
for ( i = 0; i < edrNumErrors(); i = i+1 )
{
  logStdout( "ERROR " + longToStr(i) + ": " + \
  edrGetError(i) + "\n" );
}
```

edrNumTokens

This function accesses the number of tokens attached to the current EDR container.

Syntax

```
Long edrNumTokens();
```

Parameters

None.

Return Values

Returns the number of attached tokens (this number will be greater than or equal to 0) if the function is successful. Returns **-1** if the function fails.

Example

```
for ( i = 0; i < edrNumTokens(); i = i+1 )
{
  logStdout( "Token " + longToStr(i) + ": " + \
  edrGetToken(i) + "\n" );
}
```

edrRemoveAdditionalStream

This function removes additional output streams from an EDR. Each EDR has a default output stream and any number of additional output streams.

> **Note:**
>
> This function will not remove the default output stream.

Syntax

```
Bool edrRemoveAdditionalStream(String output_stream_name);
```

Parameter

output_stream_name
The name of the output stream that you are removing from the EDR.

Return Values

Returns **true** if the function is successful or if the named stream does not exist. Returns **false** for all other conditions.

Example

This example shows how to use **edrRemoveAdditionalStream** to remove an output stream.

```
if ( edrRemoveAdditionalStream( "TELOut1" ) == false
{
logStdout( "ERROR: failed to remove additional stream: TELOut1\n" );
}
```

Example 85-1 Example removeoutmod.isc file

This example removes output module streams:

```
removeoutmod.isc
---------------
function onDetailEdr
   {
```

```
   if (edrRemoveAdditionalStream( "TelOut1" ) == true)
{
   logStdout("Stream TelOut1 removed ");
}
   if (edrRemoveAdditionalStream( "TelOut2" ) == true)
   {
      logStdout("Stream TelOut2 removed ");
   }
} // end onDetailEdr + end iScript
```

edrSetContentType

This function sets the content type of the current EDR container.

Syntax

```
Bool edrSetContentType(Long content);
```

Parameter

content
The content type to be assigned to the EDR container:

- EDR_UNKNOW_CONT
- EDR_HEADER
- EDR_DETAIL
- EDR_TRAILER
- EDR_START
- EDR_STOP
- EDR_BEGIN
- EDR_END
- EDR_BEGIN_TRANSACTION
- EDR_END_TRANSACTION

Return Values

Returns **true** if the content type is valid. Returns **false** if the container type is not valid.

Example

```
if ( edrSetContentType( EDR_TRAILER ) == false )
{
  logFormat( "ERROR: edrSetContentType() failed" );
}
```

edrSetCurrent

This function sets the current EDR container. All EDR container functions only access the current EDR container.

Syntax

```
Bool edrSetCurrent(Long index);
```

Parameter

index
The index of the EDR container you want to set. This is the return value from **edrDuplicate**.

Return Values

Returns **true** if there is an EDR container with the specified index. Returns **false** if there is no EDR container with that index.

Example

```
Long index = edrDuplicate();
if ( index < 0 )
{
  logFormat( "ERROR: duplication of edr failed" );
}
else
{
  // Set the output stream for the old container
  edrSetStream( "OrigOutput" );

  // Set the output stream for the new container
  if ( edrSetCurrent( index ) == true )
  {
    edrSetStream( "NewOutput" );
  }
}
```

edrSetIsValidDetail

This function sets the EDR container's valid detail flag. The valid detail flag specifies whether the EDR container is to be discarded.

Syntax

```
Void edrSetIsValidDetail(Bool flag);
```

Parameter

flag
The valid detail flag for the EDR container.

Return Values

Returns nothing.

Example

```
if ( ... )
{
  // record shall be discarded
  edrSetIsValidDetail( false );
}
```

edrSetStream

This function sets the output stream for an EDR. Internally, Pipeline Manager uses stream numbers instead of stream names. For this reason, the name specified must be converted to a number. If you use a constant as the stream name, the conversion can be performed at compile time, resulting in quicker performance than using a stream name that is not a constant. The second advantage of using a constant is that the existence of the stream can be checked at compile time.

> **Caution:**
>
> Illegal stream names lead to compilation errors.

Syntax

```
Bool edrSetStream(String streamName);
```

Parameter

streamName
The name of the output stream for the EDR container.

Return Values

Returns **true** if the output stream is successfully set. Returns **false** if the output stream does not exist.

Example

```
// This is the FAST method: The stream number can be evaluated \
   at compile time.
// There is also a check if the stream exists at compile time.
if ( edrSetStream( "NationalOutput" ) == false )
{
  logFormat( "ERROR: edrSetStream() failed" );
}

// This is the SLOW method and should be avoided.
String nationalOutput = "NationalOutput"

if ( edrSetStream( nationalOutput ) == false )
{
  logFormat( "ERROR: no stream " + nationalOutput );
}
```

edrString

This function retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values. When setting string values, use this function as the left-hand value in an assignment statement.

Syntax

```
String edrString(EdrField field [, Long idx1 [, Long idx2 ...]]);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the string value of the EDR field if the function is successful. Returns an empty string if the path through the EDR tree structure is not valid.

Example

```
if ( edrString( DETAIL.RECORD_TYPE) == "020" )
edrString(DETAIL.RECORD_TYPE) = "021";
```

edrStringEx

This function retrieves and sets string values in the current EDR container. This function is usually used to retrieve string values. When setting string values, use this function as the left-hand value in an assignment statement.

Syntax

```
String edrStringEx(String field, Long indicesArray, Long numIndices);
```

Parameters

field
The name of the EDR field you want to access.

indicesArray
Array of additional index values specifying the path through the EDR tree structure.

numIndices
Number of indices.

Return Values

Returns the string value of the EDR field if the function is successful. Returns an empty string if the path through the EDR tree structure is not valid.

Example

```
Long indicesArray [ ];
Long numberOfIndices;
String edrFieldName;

edrFieldName = "DETAIL.RECORD_TYPE";
numberOfIndices=0;

if ( edrStringEx(edrFieldName, indicesArray, numberOfIndices) == "020" )
edrStringEx(edrFieldName, indicesArray, numberOfIndices) = "021";
```

edrTokenString

This function retrieves the content of each token, as identified by their indexes. When the index is not available, as for a function call with no argument, this function returns the complete byte string attached to the EDR. The byte string corresponds to the original input string that generated the EDR.

The function works only when the EDR field is associated with a token through either the **edrInputMap** or **edrConnectToken** function.

Syntax

```
String edrTokenString([Long idx]);
```

Parameter

idx
The index of the token whose index you want to retrieve, where 0 <= *idx* < **edrNumTokens**.

Return Values

Returns the contents of the tokens if the function is successful. Returns an empty string if the index is invalid or there are no tokens associated with the EDR.

Example

```
logStdout( "The original (input) record corresponding to this \
EDR is \n" + edrTokenString() );
```

iRulesModeOn

This function enables the iRules mode. In the iRules mode, the init section does not consider the specified indices for an EDR field.

Syntax

```
iRulesModeOn();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
INIT_SCRIPT:
function testPrint
{
iRulesModeOff();
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,1));
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,2));
iRulesModeOn();
}
```

iRulesModeOff

This function disables the iRules mode. Disabling iRules mode ensures that the INIT takes the specified indices.

Syntax

```
iRulesModeOff();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
INIT_SCRIPT:
function testPrint
{
iRulesModeOff();
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,1));
logFormat("hyewons era hardc
-->"+edrString(DETAIL.CUST_A.PRODUCT.ERA.PA.KEY,0,0,0,2));
iRulesModeOn();
}
```

pipelineName

This function retrieves the name of the pipeline in which the script is running.

Syntax

```
String pipelineName();
```

Parameters

None.

Return Values

Returns the pipeline name.

Example

```
logPipeline("This script runs in pipeline " + pipelineName());
```

stopPipeline

This function stops the pipeline from which it is called. After the pipeline is stopped, the operator must restart the pipeline using the **ifw** command.

> **Note:**
>
> This function does not work within the BEGIN function because the pipeline object instantiation is not completed when the BEGIN function is executed.

> **Note:**
>
> Use this function only when there is an unrecoverable error that requires operation intervention.

Syntax

```
Void stopPipeline();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
if (unrecoverableError())
{
stopPipeline();
}
```

File Manipulation Functions

Table 85-7 contains file manipulation functions.

Table 85-7 File Manipulation Functions

Function	Description
fileClose	Closes a file that was opened earlier using the **fileOpen** function.
fileCopy	Copies a file.
fileDelete	Deletes a file.
fileEof	Checks to see whether the end of file has been reached.
fileFlush	Flushes the contents of the file buffer to disk.
fileIsOpen	Determines whether a file is currently open.
fileOpen	Opens a file for reading or writing. If the file is already open, the old file will be closed and the new file will be opened. The open mode is equivalent to the **fopen** C function.
fileReadLine	Reads a line from the input file. The line is read until the function encounters an end-of-line or end-of-file character or until *maxLen* is reached.
fileRename	Renames a file.

Table 85-7 (Cont.) File Manipulation Functions

Function	Description
fileSeek	Sets the read/write pointer on a specific position (in bytes from the beginning of the file) in an opened file.
fileTell	Retrieves the position (measured in bytes from the start of the file) of the read/write pointer in an opened file.
fileWriteLong	Writes a Long value, as a string and not in binary mode, to the output file.
fileWriteStr	Writes a string to the output file.

fileClose

This function closes a file that was opened earlier using the **fileOpen** function.

Syntax

```
Void fileClose(File file);
```

Parameter

file
The file you want to close.

Return Values

Returns nothing.

Example

```
File out;
if ( fileOpen( out, "test.txt", "w" ) == true )
{
  fileWriteStr( out, "Hello World!" );
  fileClose( out );
}
```

fileCopy

This function copies a file.

Syntax

```
Bool fileCopy(String old, String new);
```

Parameters

old
The file name of the file to be copied.

new
The file name of the copy.

Return Values

Returns **true** when a file has been copied. Returns **false** when it has not been copied.

Example

```
if ( fileCopy(tempName, realname ) == false )
{
logStdout( "Failed to copy" + tempName + " to " + realName );
}
```

fileDelete

This function deletes a file.

Syntax

```
Void fileDelete(String file);
```

Parameter

file
The name of the file you want to delete.

Return Values

Returns **true** if the file was successfully deleted. Returns **false** if the function failed.

Example

```
if ( fileDelete( "test.txt" ) == false )
{
  logFormat( "ERROR: failed to delete 'test.txt'" );
}
```

fileEof

This function checks to see whether the end of file has been reached.

Syntax

```
Bool fileEof(File file);
```

Parameter

file
The file you want to check.

Return Values

Returns **true** if the end of the file was reached or if no file was open. Returns **false** if it does not reach the end of the file.

Example

```
while ( fileReadLine( in, line, 2048 ) == true )
{
  ...
}
if ( fileEof( in ) == false )
{
  logFormat( "ERROR: read error()" );
}
```

fileFlush

This function flushes the contents of the file buffer to disk.

Syntax

```
Bool fileFlush(File file);
```

Parameter

file
The file you want to flush.

Return Values

Returns **true** if the file was successfully flushed. Returns **false** if the function failed.

Example

```
fileWriteStr( out, "Price is " + price );
if ( fileFlush( out ) == false )
{
  logFormat( "ERROR: fileFlush() failed" );
}
```

fileIsOpen

This function determines whether a file is currently open.

Syntax

```
Bool fileIsOpen(File file);
```

Parameter

file
The name of file you want to check.

Return Values

Returns **true** if the file is open. Returns **false** if the function failed.

Example

```
if ( fileIsOpen( in ) == false )
{
  logFormat( "ERROR: file is not open" );
}
```

fileOpen

This function opens a file for reading or writing. If the file is already open, the old file will be closed and the new file will be opened. The open mode is equivalent to the **fopen** C function.

Syntax

```
Bool fileOpen(File file, String fileName, String openMode);
```

Parameters

file
The file you want to open.

fileName
The name of the file you want to open.

openMode
The string specifying the open mode. Specify this parameter as you would for the **fopen** C function. The following description of open mode is from the Linux ManPage:

- **r:** Open text file for reading. The stream is positioned at the beginning of the file.
- **r+:** Open for reading and writing. The stream is positioned at the beginning of the file.
- **w:** Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.
- **w+:** Open for reading and writing. The file is created if it does not exist; otherwise it is truncated. The stream is positioned at the beginning of the file.
- **a:** Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file.
- **a+:** Open for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file.

Return Values

Returns **true** if the file was opened successfully. Returns **false** if the function failed.

Example

```
File out;

if ( fileOpen( out, "test.txt", "w" ) == false )
{
  logFormat( "ERROR: fileOpen() failed" );
}
```

fileReadLine

This function reads a line from the input file. The line is read until the function encounters an end-of-line or end-of-file character or until *maxLen* is reached.

Syntax

```
Bool fileReadLine(File file, String line, Long maxLen);
```

Parameters

file
The name of file you want to read.

line
The string that specifies the line to be read. This must be a left-hand value.

maxLen
The maximum length for the line.

Return Values

Returns **true** if the line is successfully read. Returns **false** if the function failed.

Example

```
File in;
String line;

if ( fileOpen( in, "test.txt", "r" ) == true )
{
  fileReadLine( in, line, 100 );
}
```

fileRename

This function renames a file. The new name can specify a different directory, but both the old and new file must be in the same file system.

Syntax

```
Bool fileRename(String old, String new);
```

Parameters

old
The old file name.

new
The new file name.

Return Values

Returns **true** if the file is successfully renamed. Returns **false** if the function failed.

Example

```
if ( fileRename( tempName, realName ) == false )
{
  logStdout( "Failed to rename " + tempName + " to " + realName );
}
```

fileSeek

This function sets the read/write pointer on a specific position (in bytes from the beginning of the file) in an opened file.

Syntax

```
Bool fileSeek(File file, Long offset);
```

Parameters

file
The file in which you want to set a read/write pointer.

offset
The position where you want to set the read/write pointer.

Return Values

Returns **true** when setting the read/write pointer in an opened file is successful. Returns **false** when it has not been successful.

Example

```
long offset = fileTell( myfile );
if ( fileSeek(myfile, offset) == false )
{
logStdout( "could not set the file read/write pointer to " + longToStr(offset) );
}
```

fileTell

This function retrieves the position (measured in bytes from the start of the file) of the read/write pointer in an opened file.

Syntax

```
Long fileTell(File file);
```

Parameter

file
The file to check.

Return Values

Returns the position of the read/write pointer when successful. Returns **(-1)** when an error occurs.

Example

```
long offset = fileTell( Myfile );
if ( offset != (-1) )
{
logStdout( "the read pointer is currently on position " + longToStr() + " to " +
realName );
}
```

fileWriteLong

This function writes a Long value to the output file. The Long value is written as a string and not in binary mode.

Syntax

```
Bool fileWriteLong(File file, Long value [, Long len [, Bool leading [, String
pad]]]);
```

Parameters

file
The file you want to write the Long value to.

value
The Long value to write.

len
The length of the output.

leading
Specifies whether to add leading or trailing characters: **true** adds leading characters, **false** adds trailing characters.

pad
The padding character to use as the first character of the string.

Return Values

Returns **true** if the Long value is successfully written. Returns **false** if the function failed.

Example

```
File out;

if ( fileOpen( out, "test.txt", "w" ) == true )
{
  fileWriteLong( out, 100, 14, true, "0" );
}
```

fileWriteStr

This function writes a string to the output file. The string is not automatically terminated by an end-of-line character.

Syntax

```
Bool fileWriteStr(File file, String string);
```

Parameters

file
The file you want to write the string to.

string
The string to write.

len
The length of the output. This parameter is optional.

leading
Specifies whether to add leading or trailing characters: **true** adds leading characters, **false** adds trailing characters.

pad
The padding character to use as the first character of the string.

Return Values

Returns **true** if the string is successfully written. Returns **false** if the function failed.

Example

```
File out;

if ( fileOpen( out, "test.txt", "w" ) == true )
{
  fileWriteStr( out, "Hello World!\n" );
}
```

Flist Manipulation Functions

Table 85-8 contains flist manipulation functions.

Table 85-8 Flist Manipulation Functions

Function	Description
fListToString	Returns the content of the current flist in string format.
fListFromString	Replaces the current flist with an flist based on the input string.
fListCount	Counts the number of elements at the top level of the current flist.
fListCreateNew	Replaces the current flist with an empty flist.
fListDate	Retrieves the date value from the current flist.
fListDecimal	Retrieves the decimal value from the current flist.
fListDropElem	Removes an array from the current flist.
fListDropFld	Deletes a field from the current flist.
fListElemid	Retrieves the array element ID from the specified array field.
fListGetErrorText	Puts the field name from the flist into *string1* and the error text into *string2.*
fListLong	Retrieves the long value from the current flist.
fListNumElem	Counts the number of elements in an array in the current flist.
fListPopElem	Resets the array to the previous value.
fListPushElem	Creates and sets the array element into which other functions set field values.
fListSetDate	Sets a date field in the current flist.
fListSetDecimal	Sets a decimal field in the current flist.
fListSetLong	Sets a long value within a PIN_FLDT_INT or PIN_FLDT_EMUN field in the current flist.
fListSetPoid	Sets a POID field in the current flist.
fListSetString	Sets a string field in the current flist.
fListString	Retrieves the string value from the current flist.
opcodeExecuteInternal	Calls the opcode specified in the parameter.

fListToString

This function returns the content of the current flist in string format. The function calls PIN_FLIST_TO_STR.

Syntax

```
String fListToString();
```

Parameters

None.

Return Values

Returns the content of the current flist in string format. Returns an empty string on failure.

Example

```
logStdout(fListToString());
fListCreateNew();
```

fListFromString

This function removes the current flist and replaces it with an flist based on a string that you pass in as a parameter. The function calls PIN_STR_TO_FLIST.

Syntax

```
Bool fListFromString(const String flist_str);
```

Parameter

flist_str
The contents of the flist to be created, in string format.

Return Values

Returns **true** on success and **false** on failure.

Example

```
String flistStr =

"0    PIN_FLD_ARRAY          ARRAY [0] allocated 13, used 1" +
"1    PIN_FLD_STRING         STR [0] \"testing\"" +
"1    PIN_FLD_DECIMAL        DECIMAL [0] 0.000" +
"1    PIN_FLD_INT            INT [0] 60";

if(!fListFromString(flistStr))
{
// flist could not be parsed
}
```

fListCount

This function counts the number of elements at the top level of the current flist by calling PIN_FLIST_COUNT.

Syntax

```
Long fListCount();
```

Parameters

None.

Return Values

Returns the number of elements at the top level of the current flist. Returns **-1** on failure.

Example

```
Long resultCounts = fListCount();
```

fListCreateNew

This function removes the current flist and replaces it with an empty flist.

Syntax

```
Bool fListCreateNew();
```

Parameters

None.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListCreateNew();
```

fListDate

This function retrieves the date value from a PIN_FLDT_TSTAMP field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Date fListDate([const String path_field [, Long elem_id]] [,const String path_field2
[, Long elem_id] ... , ] const String field);
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the date is retrieved.

Return Values

Returns the date value from the specified PIN_FLDT_TSTAMP field. Returns INVALID_DATETIME on failure.

Example

```
fListDate("PIN_FLD_RESULTS",1,"PIN_FLD_CREATED_T");
```

fListDecimal

This function retrieves the decimal value from a PIN_FLDT_DECIMAL field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Decimal fListDecimal([const String path_field [, Long elem_id]] [,const String
path_field2 [, Long elem_id] ... , ] const String field);
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the decimal value is retrieved.

Return Values

Returns the decimal value from the specified PIN_FLDT_DECIMAL field. Returns INVALID_DECIMAL on failure.

Example

```
fListDecimal("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6, "PIN_FLD_ORDER");
```

fListDropElem

This function removes an array from the current flist by calling PIN_FLIST_ELEM_DROP.

Syntax

```
Bool fListDropElem(const String array_field [,Long = 0 elem_id]);
```

Parameters

array_field
The name of the array.

elem_id
The array's element ID. The default is **0**.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListDropElem("PIN_FLD_ARGS", 2);
```

fListDropFld

This function deletes a field from the current flist by calling PIN_FLIST_FLD_DROP.

Syntax

```
Bool fListDropFld(const String field)
```

Parameter

field
The name of the field to be deleted.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListDropFld("PIN_FLD_LABEL");
```

fListElemid

This function retrieves the array element ID from the specified array field using a 0-n index in the array.

Syntax

```
Decimal fListElemid([const String path_field [, Long elemid]]
                   [,const String path_field2 [, Long elemid]
                    ... , ] const String array_field, Long index);
```

Parameters

path_field
A parent substruct or array field that is part of the path to the target array. The parameter is repeated in the case of nested arrays.

elem_id
The element ID of a parent array or substruct.

field
The name of the array from which the element ID is retrieved.

index
The 0-n index of the exact array element, the ID of which to return.

Return Values

Returns the *elem_id* value of the array element specified by 0-n index. Returns **INVALID_ARRAY** on failure.

Example

```
fListElemid("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 0);
```

fListGetErrorText

This function puts the field name from the flist into *string1* and the error text into *string2.* You can use the error information for logging or other purposes.

Syntax

```
Void fListGetErrorText(String string1, String string2);
```

Parameters

string1
String field into which the field name is placed.

string2
String field into which the error text is placed.

Return Values

Returns nothing.

Example

```
// Opcode failed
   String s1;
   String s2;
   fListGetErrorText(s1, s2);
```

fListLong

This function retrieves the long value from a PIN_FLDT_INT or PIN_FLDT_ENUM field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Long fListLong([const String path_field [, Long elem_id]] [,const String path_field2
[, Long elem_id] ... , ]const String field) ;
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the long value is retrieved.

Return Values

Returns the long value from the specified PIN_FLDT_INT or PIN_FLDT_ENUM field. Returns 0 on error.

Example

```
fListLong("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6, "PIN_FLD_LENGTH")
```

fListNumElem

This function counts the number of elements in a PIN_FLD_ARRAY field by calling PIN_FLIST_ELEM_COUNT. If the array is stored in substructs or other arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
Long fListNumElem([const String path_field [, Long elem_id]] [,const String
path_field2 [, Long elem_id] ... , ] const String array_field, Long elem_id);
```

Parameters

path_field
A substruct or array field that is part of the path to the target array. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

array_field
The name of the array.

Return Values

Returns the number of elements in the specified array. Returns **-1** on failure.

Example

```
Long resultCounts = fListNumElem("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6);
```

fListPopElem

This function resets the array to the previous value.

Syntax

```
Void fListPopElem();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
fListPopElem();
```

fListPushElem

This function creates and sets the array element into which other functions set field values. The function calls PIN_FLIST_ELEM_ADD.

Syntax

```
Bool fListPushElem(const String array_field [,Long = 0 element]);
```

Parameters

array_field
The name of the array to set.

element
The array's element ID. The default is **0**.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListPushElem("PIN_FLD_ARGS", 2);
```

fListSetDate

This function sets a date field in the current flist.

Syntax

```
Bool fListSetDate(const String field, Date value);
```

Parameters

field
The name of the date field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
Date d = strToDate("20060402143600"); // Apr 2, 2006 2:36 pm
fListSetDate("PIN_FLD_EFFECTIVE_T", d);
```

fListSetDecimal

This function sets a decimal field in the current flist.

Syntax

```
Bool fListSetDecimal(const String field, Decimal value);
```

Parameters

field
The name of the decimal field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListSetDecimal("PIN_FLD_DECIMAL",edrDecimal(DETAIL.ASS_DATA.VALUE,1));
```

fListSetLong

This function sets a long value in a PIN_FLDT_INT or PIN_FLDT_ENUM field in the current flist.

Syntax

```
Bool fListSetLong(const String field, Long value);
```

Parameters

field
The name of the long field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListSetLong("PIN_FLD_INT",edrLong(DETAIL.ASS_DATA.QUANTITY, 1));
```

fListSetPoid

This function sets a POID field in the current flist.

Syntax

```
Bool fListSetPoid(String field, String poid);
```

Parameters

field
The name of the POID field to set.

poid
The POID string to be set in the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
Bool success = fListSetPoid( "PIN_FLD_POID", "0.0.0.1 /account 1099832 0" );
```

fListSetString

This function sets a string field in the current flist.

Syntax

```
Bool fListSetString(const String field, String value);
```

Parameters

field
The name of the string field to set.

value
The value to set for the field.

Return Values

Returns **true** on success and **false** on failure.

Example

```
fListSetString("PIN_FLD_USAGE_TYPE", usageClass);
```

fListString

This function retrieves the string value from a PIN_FLDT_STR or PIN_FLDT_POID field in the current flist. If the field is stored in substructs or arrays, you must specify the path. You must include element IDs for all arrays.

Syntax

```
String fListString([const String path_field [, Long elem_id]] [,const String
path_field2 [, Long elem_id] ... , ] const String field);
```

Parameters

path_field
A substruct or array field that is part of the path to the target field. The parameter is repeated in the case of nested fields.

elem_id
The element ID of an array.

field
The name of the field from which the string value is retrieved.

Return Values

Returns the string value from the specified PIN_FLDT_STR or PIN_FLDT_POID field. Returns NULL_STRING on failure.

Example

```
fListString("PIN_FLD_OBJ_DESC", 0, "PIN_FLD_OBJ_ELEM", 6, "PIN_FLD_DESCR")
```

opcodeExecuteInternal

This function calls the opcode specified in the parameter. You can call any opcode.

You use this function in iScripts that run in a real-time pipeline. The function uses the Connection Manager (CM) context information in the EDR to call the opcode through the existing connection.

See "opcodeExecute" for information about calling opcodes in batch pipelines.

Before calling **opcodeExecuteInternal**, you compose the input flist by using the flist extension functions. The input flist is stored and used internally by the opcode call.

The output flist of the opcode call is also stored internally and replaces the input flist. It can be retrieved by using the flist extension functions again.

If there is an error in the opcode call, an error buffer will be set. The error text can be retrieved with the **fListGetErrorText**function. The error text can then be logged.

Syntax

```
Bool opcodeExecuteInternal(Long opcode, Long flags);
```

Parameters

opcode
The opcode number of the opcode to be executed.

flags
The opcode flag value. Flag values differ from opcode to opcode. Some opcodes do not expect a flag value. Use **0** for opcodes that do not expect a flag value.

Return Values

Returns **true** on success and **false** on failure.

Example

```
Long PCM_OP_SEARCH = 7;
...
if ( opcodeExecuteInternal(PCM_OP_SEARCH, 0) == false )
....
```

Hash and Array Functions

Table 85-9 contains hash and array functions.

Table 85-9 Hash and Array Functions

Function	Description
arrayClear	Clears an array.
arraySize	Determines the size of an array.
hashClear	Clears a hash.
hashContains	Checks to determine whether a hash-array contains a specific value.
hashKeys	Retrieves all keys used in an associative array.
hashRemove	Removes an entry from an associative array.

arrayClear

This function clears an array.

Syntax

```
Void arrayClear(Array array);
```

Parameter

array
The array you want to clear.

Return Values

Returns nothing.

Example

```
if ( arraySize( array ) > 0 )
{
  // Cleanup the array
  arrayClear( array );
}
```

arraySize

This function determines the size of an array.

Syntax

```
Long arraySize(Array array);
```

Parameter

array
The array whose size you want to determine.

Return Values

Returns the size of the array.

Example

```
for ( i = 0; i < arraySize( array ); i = i + 1 )
{
  logStdout( "array[" + longToStr(i) + "] = " + array[i] );
}
```

hashClear

This function clears a hash.

Syntax

```
Void hashClear(Hash hash);
```

Parameter

hash
The hash you want to clear.

Return Values

Returns nothing.

Example

```
// Cleanup the hash
hashClear( hash );
```

hashContains

This function checks to determine whether a hash-array contains a specific value.

Syntax

```
Void hashContains(Hash hash, String key);
```

Parameters

hash
The hash you want to search.

key
The value you want to search for.

Return Values

Returns **true** if the hash contains the value specified by *key*. Returns **false** if the hash does not contain this value.

Example

```
if ( hashContains( hash, "Hamburg" ) == true )
{
  logStdout( "The hash contains a value for 'Hamburg'" );
}
```

hashKeys

This function retrieves all keys used in an associative array.

Syntax

```
Long hashKeys(Hash hash, Array key);
```

Parameters

hash
The hash you want to search, looking for the key.

key
The string array as a return buffer for the keys.

Return Values

Returns the number of elements in the hash.

Example

```
String keys[];
Long age{};
Long i;

age{"Mary"} = 23;
age{"John"} = 18;

Long entries = hashKeys( age, keys );
for ( i = 0; i < entries; i = i+1 )
{
  logStdout( "Age of " + keys[i] + " is " + \
  longToStr( age{keys[i]} ) + "\n" );
}
```

hashRemove

This function removes an entry from an associative array.

Syntax

```
Bool hashRemove(Hash hash, String key);
```

Parameters

hash
The hash from which you want to remove the entry.

key
The entry to remove.

Return Values

Returns **true** if the element was removed successfully. Returns **false** if the function failed.

Example

```
if ( hashRemove( hash, "Hamburg" ) == true )
{
  logStdout( "The entry 'Hamburg' was removed from the hash\n" );
}
```

Mapping Functions

Table 85-10 contains mapping functions.

Table 85-10 Mapping Functions

Function	Description
longDecode	Maps Long values to other Long values.
strDecode	Maps string values to other string values.

longDecode

This function maps Long values to other Long values.

Syntax

```
Long longDecode(Long toMap, Long default [[, const Long src1, const Long dest1] ...]);
```

Parameters

toMap
The Long value to map.

default
The default return value if no valid mapping entry exists.

src1
The source value of the first mapping entry; this value must be a constant.

dest1
The destination value of the first mapping entry; this value must be a constant.

Return Values

Returns the matching destination value if the destination exists. Returns the value you specified in the *default* parameter if there is no destination.

Example

```
newRecordType = longDecode( oldRecordType, C_defaultRecordType,
C_oldDetail, C_newDetail,
C_oldHeader, C_newHeader,
C_oldTrailer, C_newTrailer );
```

strDecode

This function maps string values to other string values.

Syntax

```
String strDecode(String toMap, String default [[, const String src1, const String
dest1] ...]);
```

Parameters

toMap
The string value to map.

default
The default return value if no valid mapping entry exists.

src1
The source value of the first mapping entry; this value must be a constant.

dest1
The destination value of the first mapping entry; this value must be a constant.

Return Values

Returns the matching destination value if the destination exists. Returns the value you specified in the *default* parameter if there is no destination.

Example

```
newRecordType = strDecode( oldRecordType, C_defaultRecordType,
C_oldDetail, C_newDetail,
C_oldHeader, C_newHeader,
C_oldTrailer, C_newTrailer );
```

Opcode Calling Functions

Table 85-11 contains opcode calling functions.

Table 85-11 Opcode Calling Functions

Function	Description
opcodeExecute	Calls the specified opcode.
opcodeGetConnection	Obtains a connection from the specified connection pool.
pcmOpCatch	Calls PCM_OP, which performs the operation of the specified opcode and then returns the contents of the error buffer (**ebuf**) produced by the operation.

opcodeExecute

This function calls the opcode specified in the parameter. You can call any opcode.

You use this function to call opcodes in batch pipelines. See "opcodeExecuteInternal" for information about calling opcodes from real-time pipelines.

Before calling **opcodeExecute** the first time in an iScript, you must call **opcodeGetConnection** to get the connection from the connection pool. If the CM restarts or if the existing connection is broken, an error results. To get a new connection, add more conditional checks for **opcodeExecute** and then call **opcodeGetConnection**.

For example:

```
.....
Bool connectionOpened;
Long PCM_OP_NUMBER = 200;
function onBeginEdr
{
  connectionOpened = false;
}
function getCMConnection
{
 if (connectionOpened == false)
 {
 {String connectionPool = "ifw.DataPool.CMConnectionPool.Module";   connectionOpened =
opcodeGetConnection(connectionPool);
 }
}
function Bool callOpcode
{
   Long retryCount;
   Bool success;
   Long numberOfRetries = 10;
   String fldName;
   String errMsg;

getCMConnection();
success = opcodeExecute(PCM_OP_NUMBER, 0);
if (success == false)
{
   fListGetErrorText (fldName, errMsg);
   if (errMsg == "PIN_ERR_CONNECTION_LOST")
   {
    connectionOpened = false;
    for (retryCount = 0; ((retryCount < numberOfRetries) and (connectionOpened ==
false)); retryCount = retryCount + 1)
    {
        connectionOpened = false
        getCMConnection();
        if(connectionOpened == true)
        {
           success = opcodeExecute(PCM_OP_NUMBER,0);
         }
    if ((connectionOpened == false) and (retryCount >= numberOfRetries))
    {
     logStdout("Error executing opcode PCM_OP_GET_PIN_VIRTUAL_TIME due to lost
connection with CM\n");
    }
     if ((success == false)
      {
        logStdout("Error: "+ errMsg + "while executing opcode
PCM_OP_GET_PIN_VIRTUAL_TIME\n");
       }
      return success;
    function  onDetailEdr()
    {
      Bool success = callOpcode()
    }
.....
```

Before calling **opcodeExecute**, you compose the input flist by using the flist extension functions. The input flist is stored and used internally by the opcode call.

The output flist of the opcode call is also stored internally and replaces the input flist. It can be retrieved by using the flist extension functions again.

If there is an error in the opcode call, an error buffer will be set. The error text can be retrieved with the **fListGetErrorText** function. The error text can then be logged.

Syntax

```
Bool opcodeExecute(Long opcode, Long flags);
```

Parameters

opcode
The opcode number of the opcode to be executed.

flags
The opcode flag value. Flag values differ from opcode to opcode. Some opcodes do not expect a flag value. Use **0** for opcodes that do not expect a flag value.

Return Values

Returns **true** on success and **false** on failure.

Example

```
...
Long PCM_OP_SEARCH = 7;
Bool success = opcodeExecute(PCM_OP_SEARCH, 0)
...
```

opcodeGetConnection

This function obtains a connection to the CM from the specified connection pool in a batch pipeline. You must configure a connection pool in the pipeline before using this function. See *DAT_ConnectionPool* in the BRM documentation for information about configuring a connection pool.

In an iScript, you must call **opcodeGetConnection** before calling **opcodeExecute** the first time. You do not need to call **opcodeGetConnection** again for subsequent opcode calls in the same script. Adding more conditional checks ensures that **opcodeGetConnection** is not called every time a CDR is processed.

> **Note:**
>
> This function is required in iScripts used in batch pipelines only. It is not necessary in real-time pipelines.

For example:

```
........
Bool connectionOpened;
function onBeginEdr
{
connectionOpened = false;
}
function getCMConnection
{
```

```
  if(connectionOpened == false)
   {
   String connectionPool = "ifw.DataPool.CMConnectionPool.Module";
   connectionOpened = opcodeGetConnection(connectionPool);
   }
  if(connectionOpened == false)
   {
      logStdout("Unable to get connection to CM\n");
   }
}
.........
```

Syntax

```
Bool opcodeGetConnection(String connectionPool);
```

Parameter

connectionPool
The full registry name of the connection pool used for the pipeline.

Return Values

Returns **true** on success and **false** on failure.

Example

```
...
String connectionPool = "ifw.DataPool.CMConnectionPool.Module";
Bool success = opcodeGetConnection(connectionPool);
...
```

pcmOpCatch

This function calls the PCM_OP opcode, which performs the operation of the specified opcode and then returns the contents of the error buffer (**ebuf**) produced by the operation. This enables the calling iScript to resolve any errors that occur during the operation without exiting the logic the iScript is performing.

Syntax

```
pcmOpCatch(opcode, flags, in_flistp, ebufp);
```

Parameters

opcode
The name or number of the opcode whose operation PCM_OP is to perform.

> **Note:**
>
> Opcode numbers are listed in the **pcm_ops.h** file in the *BRM_home***/include** directory, where *BRM_home* is the directory in which you installed the BRM components.

flags
The flags supported by the opcode being called. See the opcode description for information on the flags it takes.

- If the opcode takes no flags, enter **0**.
- To specify multiple flags, separate the flag names with a vertical bar (|).

in_flistp
A pointer to the input flist of the opcode being called. See the individual opcode flist reference pages for the input flist specifications.

ebufp
A pointer to the error buffer that stores any errors that occur during the operation.

Return Values

This function returns nothing.

Errors are passed back to the calling iScript through the specified error buffer.

Example

```
pcmOpCatch(7, SRCH_DISTINCT, search, PINERR);
```

Pipeline System Functions

Table 85-12 contains Pipeline system functions.

Table 85-12 Pipeline System Functions

Function	Description
formatName	Determines the name of the format the script is running in.
logFormat	Writes messages to the pipeline log
logPipeline	Writes messages to the pipeline log.
msgArg	Deprecated.
msgName	Deprecated.
msgNumArgs	Deprecated.
registryNodeName	Returns the name of the registry node in which the script (iScript or input/output grammar) is running.
regString	Retrieves values from the registry.
reqSend	Sends a request to a registered object and waits for an answer (i.e., synchronous messaging).
scriptUsable	Sets the *usable* flag for the script. If the *usable* flag is set to **false** in the BEGIN function during Pipeline Manager startup, Pipeline Manager will not start to process CDRs. The **false** setting can be useful if the iScript initialization fails.
sendEvent	Sends an event to the event handler.
stopFormat	Stops the format; for example, after critical errors.

formatName

This function determines the name of the format the script is running in.

Syntax

```
String formatName();
```

Parameters

None.

Return Values

Returns the format name.

Example

```
logFormat( "This script runs in format " + formatName() );
```

logFormat

This function writes messages to the pipeline log.

> **Note:**
>
> This function is obsolete and should be replaced by the **logPipeline** function.

Syntax

```
Void logFormat(String msg);
```

Parameter

msg
The message to write to the pipeline log.

Return Values

Returns nothing.

Example

```
logFormat( "Hello World!" );
```

logPipeline

This function writes messages to the pipeline log.

Syntax

```
Void logPipeline(String msg [, Long severity]);
```

Parameters

msg
The message to write to the pipeline log.

severity
The severity of the message:

- **0** = Debug
- **1** = Normal
- **2** = Warning
- **3** = Minor error
- **4** = Major error
- **5** = Critical error

  The default is **0**.

Return Values

Returns nothing.

Example

```
logPipeline( "ERROR: critical database error occurred", 4 );
```

registryNodeName

This function returns the name of the registry node in which the script (iScript or input/output grammar) is running.

Syntax

```
String registryNodeName();
```

Parameters

None.

Return Values

Returns the name of the registry node in which the script (iScript or input/output grammar) is running.

Example

```
logFormat( "This script is located at registry: " + registryNodeName () );
//this will return the following result,
//This script is located at registry:
ifw.Pipelines.ciber25.Functions.Thread1.FunctionPool.myIScript.Module.Scripts.retrieve
```

regString

This function retrieves values from the registry.

Syntax

```
String regString(String name);
```

Parameter

name
The name of the registry entry.

Return Values

Returns the specified registry entry if it exists. Returns an empty string if there is no registry entry with that name.

Example

```
if ( regString( "IntegRate.DataPool.Customer.Module.Source" ) ==\
"FILE" )
{
  logFormat( "Customers are read from file" );
}
```

reqSend

This function sends a request to a registered object and waits for an answer (i.e., synchronous messaging).

Syntax

```
Bool reqSend(String reqDestination, String reqName, Array inParams, Array outParams);
```

Parameters

reqDestination
The registry name of the request's destination.

reqName
The name of the request.

inParams
A string array containing the input parameter expected by the destination to be able to process the request.

outParams
A string array to contain the reply to the request.

Request Names

REQ_NEWSEQUENCENUMBER
(Sequencer) Returns the new sequence number.

REQ_CC
(Pipeline) Returns the country code defined in the registry for this pipeline.

REQ_MCC
(Pipeline) Returns the mobile country code defined in the registry for this pipeline.

REQ_NAC
(Pipeline) Returns the national access code value defined in the registry for this pipeline.

REQ_IAC
(Pipeline) Returns the international access code defined in the registry for this pipeline.

REQ_IAC_SIGN
(Pipeline) Returns the international access code sign value defined in the registry for this pipeline.

REQ_NDC
(Pipeline) Returns the national destination code value defined in the registry for this pipeline.

REQ_REJECT_STREAM_NAME
(Pipeline) Returns the reject stream name defined in the registry for this pipeline.

REQ_REJECT_STREAM
(Pipeline) Returns the reject stream number defined in the registry for this pipeline.

REQ_EVENTHANDLER_NAME
(ifw) Returns the event handler name.

REQ_ERROR_FILENAME
(Input) Returns the name and path of the error file.

REQ_INPUT_FILENAME
(Input) Returns the name and path of the input file.

REQ_INPUT_TEMP_FILENAME
(Input) Returns the name and path of the temporary input file.

REQ_DONE_FILENAME
(Input) Returns the name and path of the done file.

REQ_RETURN_FILENAME
(Input) Returns the name and path of the return file.

REQ_OUTPUT_FILENAME
(Output) Returns the name and path of the output file.

REQ_OUTPUT_TEMP_FILENAME
(Output) Returns the name and path of the temporary output file.

Return Values

Returns **true** if the request has been sent and an answer received successfully. Returns **false** if sending the request has failed.

Example

```
sendArray [0] = "abcdefg.so142" ;
if ( reqSend( reg_InputStream, "REQ_ERROR_FILENAME",sendArray, receiveArray)==true )
{
String errFileName = receiveArray[0]; // the fully qualified filename (including path)
}
```

scriptUsable

This function sets the *usable* flag for the script. If the *usable* flag is set to **false** in the BEGIN function during Pipeline Manager startup, Pipeline Manager will not start to process CDRs. The **false** setting can be useful if the iScript initialization fails.

> **Note:**
>
> You can use this function only in the iScript modules and not in the input grammar.

Syntax

```
Void scriptUsable(Bool usable);
```

Parameter

usable
The flag indicating whether the script is usable.

Return Values

Returns nothing.

Example

```
function BEGIN
{
  ...
  if ( fileOpen( inFile, "data.txt", "r" ) == false )
  {
    logFormat( "failed to open data file 'data.txt'" );
    scriptUsable( false );
  }
}
```

sendEvent

This function sends an event to the event handler.

Syntax

```
Bool sendEvent(String event [, String arg1 [, String arg2 ...]]);
```

Parameters

event
The name of the event to send.

argX
A comma-delimited number of argument strings used as parameters for the event.

Return Values

Returns **true** if the event was successfully sent. Returns **false** if the function failed.

Example

```
if ( sendEvent( EVT_FILE_PROCESSED, filename ) == false )
{
  logFormat( "ERROR: sendEvent() failed" );
};
```

stopFormat

This function stops the format; for example, after critical errors.

Syntax

```
Void stopFormat();
```

Parameters

None.

Return Values

Returns nothing.

Example

```
if ( fileWriteString( out, data ) == false )
{
  logFormat( "ERROR: fileWriteString() failed" );
  stopFormat();
};
```

Static Functions

This section describes static functions.

EXT_ConvertCli::convert

This function normalizes wireless and wireline CLIs into international format.

Syntax

```
const BAS_String EXT_ConvertCli::convert(const BAS_String& cli,
                                         const BAS_String& modInd,
                                         const long typeOfNumber,
                                         const BAS_String& natAccessCode,
                                         const BAS_String& intAccessCode,
                                         const BAS_String& countryCode,
                                         const BAS_String& intAccessCodeSign,
                                         const BAS_String& natDestinCode );
```

Parameters

cli
CLI to normalize.

modInd
Modification Indicator, for example, "00".

typeOfNumber
Type Of Number, for example, 0.

natAccessCode
National Access Code, for example, "0".

intAccessCode
International Access Code, for example, "00".

countryCode
Country Code, for example, "49".

intAccessCodeSign
International Access Code Sign, for example, "+".

natDestinCode
National Destination Code, for example, "172".

Return Values

Returns a CLI in international normalized format: <iac>< cc><ndc>extension.

Example

```
...
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normCli;
BAS_String cli = "01721234567";

normCli = EXT_ConvertCli::convert( cli, "00", 0, "0", "00", "49", "+", "172" );

// normCli now contains: 00491721234567
...
```

EXT_ConvertIPv4::convert

This function normalizes IPv4 addresses.

Syntax

```
const BAS_String EXT_ConvertIPv4::convert( const BAS_String& ip );
```

Parameter

ip
The IP address to normalize.

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 3 digits with zeroes.

Example

```
....
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normIp;
BAS_String ip = "192.168.1.253";

normIp = EXT_ConvertIPv4::convert( ip );

// normIp now contains: 192168001253
```

```
...
```

EXT_ConvertIPv6::convert

This function normalizes IPv6 addresses.

Syntax

```
const BAS_String EXT_ConvertIPv6::convert( const BAS_String& ip );
```

Parameter

ip
The IP address to normalize

Return Values

Returns an IP address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normIp;
BAS_String ip = "0:0:0:AF:E:0:1:FE";

normIp = EXT_ConvertIPv6::convert( ip );

// normIp now contains: 00000000000000AF000E0000000100FE

...
```

EXT_ConvertIPv4onv6::convert

This function normalizes IPv4 over IPv6 addresses. The decimal IPv4 address is converted into hexadecimal representation.

Syntax

```
const BAS_String EXT_ConvertIPv4onv6::convert( const BAS_String& ip );
```

Parameter

ip
The IP address to normalize.

Return Values

Returns an IPv6 address in normalized format.

Dots (.) are skipped. Tokens are left-padded to 4 digits with zeroes.

Example

```
....
#include "EXT_ConverterExt.hpp"
#include "EXT_CliConverter.hpp"

BAS_String normIp;
BAS_String ip = "0:0:0:0:0:0:192.168.10.1";

normIp = EXT_ConvertIPv4onv6::convert( ip );

// normIp now contains: 00000000000000000000000C0A80A01

...
```

Standard Functions

Table 85-13 contains standard functions.

Table 85-13 Standard Functions

Function	Description
closeClientConnection	Closes the connection to the Diameter client
currentTimeInMillis	Gets the current system time in milliseconds.
getClientState	Gets the state of a Diameter client.
mutexAcquire	Acquires the mutex specified by the handle (a number that identifies the mutex). When the mutex specified by the handle is already acquired by another thread, the function call is blocked unless the other thread releases the mutex by calling the **mutexRelease** function.
mutexCreate	Creates a mutex that can later be accessed by its handle.
mutexDestroy	Used to destroy a mutex that is no longer needed.
mutexRelease	Releases a mutex that has been acquired. It unblocks a functional call by another thread that has been trying to acquire the mutex using the **mutexAcquire** function.
sleep	Makes the process sleep.
startTimer	Starts the timer.
sysExecute	Executes a command line in a file.
sysGetEnv	Gets an environment variable.

closeClientConnection

This function closes the connection to the Diameter client.

Syntax

Void closeClientConnection(Socket Num);

Parameter

Num
Socket Id of the Diameter client.

Return Values

Returns nothing.

Example

```
if( (commandCode == DIA_DP_REQUEST) and (commandFlag == 0) )
{
  logPipeline("CommandCode: DIA_DP_REQUEST. Closing the connection.",0);
  closeClientConnection(edrLong(DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.SOCKETID,0,0));
}
```

currentTimeInMillis

This function gets the current system time in milliseconds.

You can use this function in your custom iScript to record the time when a pipeline or a pipeline module starts processing an EDR and when it finishes processing the EDR. You can then calculate the difference between the start and end times to determine the latency of the EDR processing in a pipeline or module.

You can include the iScript at any point in a pipeline to determine the latency of an EDR processing between two points in a pipeline.

Syntax

```
Long currentTimeInMillis();
```

Parameters

None.

Return Values

Returns the current system time as a long value.

Example

This example gets the current system time and logs a message:

```
logStdout("The Time in milliseconds is = " + longToStr(currentTimeInMillis()) + "\n");
```

getClientState

This function gets the state of a Diameter client.

Syntax

Long getClientState(Socket Num);

Parameter

Num
Socket Id of the Diameter client.

Return Values

Returns one of the following state values:

- 0 = STATE_INITIAL
- 1 = STATE_OKAY
- 2 = STATE_DOWN

Example

```
state =
getClientState(edrLong(DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.SOCKETID,
0,0));
```

mutexAcquire

This function acquires the mutex specified by the handle (a number that identifies the mutex). When the mutex specified by the handle is already acquired by another thread, the function call is blocked unless the other thread releases the mutex by calling the **mutexRelease** function.

Syntax

```
Bool mutexAcquire(Long handle);
```

Parameter

handle
The handle of the mutex to acquire.

Return Values

Returns **true** if a valid handle is used and the mutex is acquired. Returns **false** if an invalid handle is used and the mutex is not acquired.

Example

```
// enter the protected area
mutexAcquire (handle)

// protected area

//leave the protected area
mutexRelease(handle)
```

mutexCreate

This function creates a mutex that can later be accessed by its handle.

Syntax

```
Long mutexCreate();
```

Parameters

None.

Return Values

Returns a handle (**>0**) if the mutex was created successfully. Returns **<0** if the mutex was not created successfully.

Example

```
long handle; function BEGIN
{
handle = mutexCreate ( )
if (handle < 0)
{
 logStdout("Mutex creation failed\n");
}
}
```

mutexDestroy

This function destroys a mutex that is no longer needed.

Syntax

```
Bool mutexDestroy(Long handle);
```

Parameter

handle
The handle of the mutex to be destroyed.

Return Values

Returns **true** when destroying the mutex is successful. Returns **false** when destroying the mutex has not been successful.

Example

```
if ( mutexDestroy (handle) == false )
{
logStdout( "Illegal mutex handle\n");
}
```

mutexRelease

This function releases a mutex that has been acquired. It unblocks a functional call by another thread that has been trying to acquire the mutex using the **mutexAcquire** function.

Syntax

```
Bool mutexRelease(Long handle);
```

Parameter

handle
The handle of the mutex you want to release.

Return Values

Returns **true** when a valid handle was used and the mutex is released successfully. Returns **false** when the handle used is invalid and the mutex is not released.

Example

```
// enter the protected area
mutexAcquire (handle)

// protected area

// leave the protected area
mutexRelease(handle)
```

sleep

This function makes the process sleep.

Syntax

```
Void sleep(Long seconds);
```

Parameter

seconds
The number of seconds you want the process to sleep.

Return Values

Returns nothing.

Example

```
sleep (10)
```

startTimer

This function starts the timer.

Syntax

Void startTimer(Socket Num);

Parameter

Num
Socket Id of the Diameter client.

Return Values

Returns nothing.

Example

startTimer(edrLong(DETAIL.ASS_PROTOCOL_INFO.ASS_DIAMETER_INFO.SOCKETID,0,0));

sysExecute

This function executes a command line in a file. When you call this function in an iScript, you must configure an EventHandler in the pipeline registry file. For example:

```
EventHandler
{
  ModuleName = EVT
  Module
  {
   Events
    {
    }
   Buffer
    {
     Size = 1000
    }
  }
}
```

Syntax

```
Long sysExecute(String commandLine [String returnBuffer, Long timeToWait]);
```

Parameters

commandLine
The command line to execute. The value must be the path to an executable, followed by any arguments.

returnBuffer
A string to collect the output produced on stdout by *commandLine*. The stdin and stderr for *commandLine* will be the terminal.

timeToWait
The maximum time (in seconds) to wait for the response from the event handler. Command execution is terminated when *timeToWait* expires.

Return Values

Returns a Long value greater than 0 if the function is successful. Returns **-1** if the specified path points to a file that is either not readable or not executable.

Example

```
// list the contents of the /data/input directory

String cmdline = "/usr/bin/ls -l /data/input";
String retbuf;
Long timeToWait = 10; // 10 seconds
Long retval = sysExecute( cmdline, retbuf, timeToWait );
if ( retval != -1 )
{
```

```
  // code to process retbuf
  logStdout( retbuf );
}
```

sysGetEnv

This function specifies an environment variable you want returned.

Syntax

```
String sysGetEnv(String envVariable);
```

Parameter

envVariable
The name of the environment variable you want returned.

Return Values

Returns the specified environment variable and its settings.

Example

```
logStdout("******************-\n");
logStdout( "PATH=" + sysGetEnv( "PATH") +"\n");directory \n");
```

String Functions

Table 85-14 contains string functions.

Table 85-14 String Functions

Function	Description
decimalToStr	Converts a decimal value into a string.
decimalToStrHex	Converts a decimal value into a hexadecimal string. Use **round(***value***)** or **trunc(***value***)** to remove the decimal portion if you do not want it to be coded in hexadecimal.
longToHexStr	Converts a Long value into a hexadecimal string.
longToStr	Converts a Long value into a string.
strByteValue	Converts the first character in the input string to its byte value.
strDecode	Maps string values to other string values.
strEndsWith	Checks to see if a string ends with a special suffix.
strHexStrToStr	Converts each pair of characters in a given hexadecimal string into the equivalent single-byte ASCII character in a new string. The returned string is half the size of the original. Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets will cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.
strHexToDecimal	Converts a hexadecimal string to a decimal value.
strHexToLong	Converts a hexadecimal string into a Long value.
strLength	Determines the length of a string.

Table 85-14 (Cont.) String Functions

Function	Description
strMatch	Compares a regular expression to a string, looking for a match.
strPad	Pads a string to a specific length. The padding character and the justification can be selected.
strReplace	Replaces substrings in a string.
strSearch	Searches for a substring inside another string.
strSearchRegExpr	Searches for a regular expression to a string.
strSplit	Splits a string according to a specific separator character and stores the resulting tokens in a string array.
strStartsWith	Checks to see if a string starts with a specified prefix.
strStrip	Removes special leading or trailing characters from a string.
strStrToHexStr	Converts each character in a given string into its two-character hexadecimal equivalent in a new string. The returned string is twice the size of the original. Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.
strSubstr	Extracts a substring from a string.
strToDate	Converts a string into a date value.
strToDecimal	Converts string values to decimal values.
strToLong	Converts a string value to a Long value.
strToLower	Converts a string to lowercase characters.
strToUpper	Converts a string to uppercase characters.

decimalToStr

This function converts a decimal value into a string.

Syntax

```
String decimalToStr(Decimal value [, Long precision]);
```

Parameters

value
The value to convert into a string.

precision
The number of digits after the decimal point.

Return Values

Returns the value as a string.

Example

```
logFormat( "Pi = " + decimalToStr(pi) );
logFormat( "Pi = " + decimalToStr(pi,2) );
```

decimalToStrHex

This function converts a decimal value into a hexadecimal string.

> **Note:**
>
> Use **round(***value***)** or **trunc(***value***)** to remove the decimal portion if you do not want it to be coded in hexadecimal. For example, use **round(0)** to omit the .000 if you want only integer values returned.

Syntax

```
String decimalToStrHex(Decimal value [, String separator [, Long precision]]);
```

Parameters

value
The decimal value to convert into a hexadecimal string. Code this in readable ASCII.

separator
The character you want to use as a decimal separator (the default is **.**).

precision
The precision of the decimal value to use when generating the hexadecimal string (the default is **0**).

Return Values

Returns the decimal value as a hexadecimal string.

Example

```
logFormat( "X = " + decimalToStr(x) + "(" + decimalToStrHex(x) + \
" hexadecimal)" );
```

longToHexStr

This function converts a Long value into a hexadecimal string.

Syntax

```
String longToHexStr(Long value);
```

Parameter

value
The Long value to convert into a hexadecimal string.

Return Values

Returns the value as a hexadecimal string.

Example

```
logFormat( "X = " + longToStr(x) + "(" + longToHexStr(x) + \
" hexadecimal)" );
```

longToStr

This function converts a Long value into a string.

Syntax

```
String longToStr(Long value);
```

Parameter

value
The Long value to convert into a string.

Return Values

Returns the value as a string.

Example

```
logFormat( "X = " + longToStr(x) );
```

strByteValue

This function converts the first character in the input string to its byte value.

Syntax

```
Long strByteValue(String string);
```

Parameter

string
The string whose first character you want to convert.

Return Values

Returns the byte value of the first character if the function is successful. Returns **0** if the string is empty.

Example

```
Long ascA = strByteValue( "A" );
logStdout( "ASCII(A) = " + longToStr( ascA ) + "\n" );
```

strDecode

This function maps string values to other string values.

Syntax

```
String strDecode(String toMap, String default [[, const String src1, const String
dest1] ...]);
```

Parameters

toMap
The string value to map.

default
The default return value if no valid mapping entry exists.

src1
The source value of the first mapping entry; this value must be a constant.

dest1
The destination value of the first mapping entry; this value must be a constant.

Return Values

Returns the matching destination value if the destination exists. Returns the value you specified in the *default* parameter if there is no destination.

Example

```
newRecordType = strDecode( oldRecordType, C_defaultRecordType,
C_oldDetail, C_newDetail,
C_oldHeader, C_newHeader,
C_oldTrailer, C_newTrailer );
```

strEndsWith

This function checks to see if a string ends with a special suffix.

Syntax

```
Bool strEndsWith(String string, String suffix);
```

Parameters

string
The string to check the suffix for.

suffix
The suffix to check.

Return Values

Returns **true** if the string ends with the specified suffix. Returns **false** if the string does not end with the suffix.

Example

```
if ( strEndsWith( filename, ".txt" ) )
{
  logFormat( "file suffix is .txt" );
}
```

strHexStrToStr

This function converts each pair of characters in a given hexadecimal string into the equivalent single-byte ASCII character in a new string. The returned string is half the size of the original. For example, if you pass the string **58595A373839** to **strHexStrToStr**, it returns the string **XYZ789**.

Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets will cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.

Syntax

```
String strHexStrToStr(source);
```

Parameter

source
The hexadecimal string to convert to ASCII:

- It must have an even number of characters.
- Only numeric characters and A through F are permitted.
- It cannot be empty.

Return Values

Returns the string converted to ASCII if the function is successful.

If *source* has hexadecimal representations for embedded nulls, the returned string contains embedded nulls. The caller must interpret such strings correctly.

Example

```
String source = ""58595A373839";
String result = strHexStrToStr(source);
logStdout(result);
```

strHexToDecimal

This function converts a hexadecimal string to a decimal value.

Syntax

```
Decimal strHexToDecimal(String string [, String separator [, Long precision]]);
```

Parameters

string
The hexadecimal string (coded in readable ASCII) to convert into a decimal value.

separator
The character you want to use as decimal separator (the default is **.**).

precision
The precision of the decimal value to be generated (the default is **0**).

Return Values

Returns a decimal value when the value entered for *string* is successfully converted to a decimal value. Returns **0.0** if *string* is not a valid hexadecimal decimal/Long value and is therefore not converted to a decimal value.

Example

```
logStdout ( "1FF hex is " + decimalToStr ( strHexToDecimal ( "1FF" ) ) + " decimal\n" );
```

strHexToLong

This function converts a hexadecimal string into a Long value.

Syntax

```
Long strHexToLong(String string);
```

Parameter

string
The hexadecimal string to convert into a Long value.

Return Values

Returns the hexadecimal string as a Long value.

Example

```
logStdout( "1FF hex is " + strHexToLong( "1FF" ) + " decimal\n" );
```

strLength

This function determines the length of a string.

Syntax

```
Long strLength(String string);
```

Parameter

string
The string whose length you want to determine.

Return Values

Returns the string length in characters if the function is successful.

Example

```
if ( strLength( edrString( DETAIL.RECORD_TYPE ) ) != 3 )
{
  logFormat( "WARNING: illegal RECORD_TYPE" );
};
```

strMatch

This function compares a regular expression to a string, looking for a match.

Syntax

```
String strMatch(String string, String regExp [, Long index]);
```

Parameters

string
The string that you want to search for the regular expression.

regExp
The regular expression to match against the string.

index
The starting index for the search; the beginning of the string has an index of 0 (the default is **0**).

Return Values

Returns the matching part of the string if the function is successful. Returns **0** if the function does not find a match.

Example

```
if ( strMatch( filename, ".*\\.edr" ) != "" ) // IMPORTANT: the first \ is removed by
the compiler!!!!
{
  logFormat( filename + " is a *.edr file" );
}
```

strPad

This function pads a string to a specific length. The padding character and the justification can be selected.

> **Note:**
>
> The original string you started with will be truncated. If the original string is greater in length than the string you set up to result from applying the **String strPad** function.

Syntax

```
String strPad(String string, String padChar, Long length, Bool isLeftJustified);
```

Parameters

string
The string to pad (or truncate) to a specified length.

padChar
The pad character to use (the first of the string is used if empty).

length
The desired length of the returned string. If *length* is less than or equal to 0, an empty string is returned.

isLeftJustified
If set to **true**, it specifies that the string be left justified. If set to **false**, it specifies that the string be right justified.

Return Values

Returns the padded or truncated string.

Example

```
String resString;
resString = strPad ("hello", " ", 2, true); // -> resString = "he";
resString = strPad ("hello", " ", 2, false); // -> resString = "he";
resString = strPad ("hello", " ", 10, true); // -> resString = "hello ";
resString = strPad ("hello", " ", 10, false); // -> resString = "  hello";
resString = strPad ("hello", "0", 10, false); // -> resString = "00000hello";
resString = strPad ("hello", " ", -2, true); // -> resString = "";
```

strReplace

This function replaces substrings in a string.

Syntax

```
String strReplace(String toReplace, Long pos, Long len, String replace);
```

Parameters

toReplace
The string in which you want the substring replaced.

> **Note:**
>
> The input string in *toReplace* is not changed.

pos
The start position of the substring to replace. Positions start with 0.

len
The length of the substring to replace.

replace
The replacement string.

Return Values

Returns a string with the replacement string in the correct position. Returns an empty string if *pos* and *len* do not specify a valid substring.

Example

```
logFormat( strReplace( "Hello !", 5, 1, "World " ) );
```

strSearch

This function searches for a substring inside another string.

Syntax

```
Long strSearch(String string, String search [, Long index]);
```

Parameters

string
The string that you want to search.

search
The string that you want to search for.

index
The starting index for the search; the beginning of the string has an index of 0 (the default is **0**).

Return Values

Returns the starting index (this should be a value greater than or equal to 0) for the search within the string. Returns a value less than 0 if the function does not find the string.

Example

```
if ( strSearch( edrString( DETAIL.B_NUMBER ), "0049", 0 ) >= 0 )
{
  logFormat( "B-Number contains '0049'" );
}
```

strSearchRegExpr

This function searches for a regular expression to a string.

Syntax

```
Long strSearchRegExpr(String string, const String regExp [, Long index]);
```

Parameters

string
The string that you want to search.

regExp
The regular expression to look for in the string.

> **Note:**
>
> The strSearchRegExpr function does not support braces ({ }); for example, A{1,3}.

index
The starting index for the search; the beginning of the string has an index of 0 (the default is **0**).

Return Values

Returns the position index (this should be a value greater than or equal to 0) of the string if the function is successful. Returns a value less than 0 if the function does not find the string.

Example

```
if ( strSearchRegExpr( filename, ".*\\.doc", 0 ) >= 0 )
// IMPORTANT: the first \ is removed by the compiler
{
  logFormat( filename + " is a *.doc file" );
}
```

strSplit

This function splits a string according to a specific separator character and stores the resulting tokens in a string array.

Syntax

```
Long strSplit(Array res, String string, String sep);
```

Parameters

res
The resulting array to fill.

string
The input string to split.

sep
The separator to use for splitting. If the separator you specify is longer than one character, the function uses only the first character.

Return Values

Returns the number of elements in the resulting array.

Example

```
String ListArray[];
String ListString;
ListArray="first,second,third"
Long nbElem = strSplit( ListArray, ListString, "," );
for (Long i=0 ; i<nbElem ; i=i+1)
{
  logStdout( "Element " + ListArray[i] + "\n");
}
```

strStartsWith

This function checks to see if a string starts with a specified prefix.

Syntax

```
Bool strStartsWith(String string, String prefix);
```

Parameters

string
The string in which to check for the specified prefix.

prefix
The specified prefix being checked for in the string.

Return Values

Returns **true** if the string starts with the specified prefix. Returns **false** if the string does not start with the specified prefix.

Example

```
if ( strStartsWith( edrString( DETAIL.B_NUMBER ), "0049" ))
{
isNationalCall = true;
}
```

strStrip

This function removes special leading or trailing characters from a string.

Syntax

```
Bool strStrip(String string [, Long stripMode [, String stripChar]]);
```

Parameters

string
The string from which you want to remove leading or trailing characters.

stripMode
The strip mode:

- STRIP_LEADING
- STRIP_TRAILING
- STRIP_BOTH

The default is **STRIP_LEADING**.

stripChar
The character to be removed, which is the first or last character of the string (the default is the space character).

Return Values

Returns the stripped string.

Example

```
String test = "--------Hello-------------";
if ( strStrip( test, STRIP_BOTH, "-" ) == "Hello" )
{
  logStdout( "strStrip() works correct" );
}
```

strStrToHexStr

This function converts each character in a given string into its two-character hexadecimal equivalent in a new string. The returned string is twice the size of the original. For example, if you pass the string **XYZ789** to **strStrToHexStr**, it returns the string **58595A373839**.

Only ASCII values from 0 through 255 can be handled by this function. Characters from multi-byte character sets cause unexpected results. The function fails if memory cannot be allocated for the string to be returned.

Syntax

```
String strStrToHexStr(source);
```

Parameter

source
The ASCII string to convert to hexadecimal. It cannot be empty. Embedded nulls are permitted and handled correctly.

Return Values

Returns the string converted to hexadecimal if the function is successful.

Example

```
String source = "XYZ789";
String result = strStrToHexStr(source);
logStdout(result);
```

strSubstr

This function extracts a substring from a string.

Syntax

```
String strSubstr(String string, Long pos, Long len);
```

Parameters

string
The string from which you want to extract the substring.

pos
The start position of the substring to extract. Positions start with 0.

len
The length of the substring to extract.

Return Values

Returns the specified string if the function is successful. Returns an empty string if *pos* and *len* do not specify a valid substring.

Example

```
if ( strLength( string ) > 6 )
{
  string = strSubstr( string, 0, 6 );
}
```

strToDate

This function converts a string into a date value. The only supported string format is YYYYMMDDHHMMSS.

Syntax

```
Date strToDate(String dateStr);
```

Parameters

%%
The literal **%** character.

%d
The day of the month; for example, 29. The range is 00-31.

%H
The hour of the 24-hour day; for example, 14. The range is 00-23.

%m
The month of the year, from 01; for example, 02. The range is 01-12.

%M
The minutes after the hour; for example, 34. The range is 00-59.

%S
The seconds after the minute; for example, 56. The range is 00-59.

%y
The year of the century, from 00; for example, 04 for 2004. The range is 01-99. In most cases, you should avoid this parameter.

%Y
The year including the century; for example, 1994.

Return Values

Returns a valid date if the input string is in the right format. Returns an invalid date if the format is not correct.

Example

```
edrDate(DETAIL.CHARGING_START_TIMESTAMP) = \
strToDate("24.12.2002", "%d. %m. %Y");
```

strToDecimal

This function converts string values to decimal values.

Syntax

```
Decimal strToDecimal(String string);
```

Parameter

string
The string to convert to a decimal value.

Return Values

Returns the string converted to a decimal value if the function is successful. Returns **0** if the string is not a valid decimal value.

Example

```
x = x + strToDecimal( "13.32" );
```

strToLong

This function converts a numeric string value to a Long value. An alphanumeric string is returned as **0**.

Syntax

```
Long strToLong(String string);
```

Parameter

string
The string to convert to a Long value.

Return Values

Returns the string converted to a Long value if the function is successful. Returns **0** if the string is not a valid Long value.

Example

```
if ( strToLong( edrString(DETAIL.RECORD_TYPE) ) == 20 )
{
  // Basic detail record
}
```

strToLower

This function converts a string to lowercase characters.

Syntax

```
String strToLower(String string);
```

Parameter

string
The string to convert to lowercase characters.

Return Values

Returns the string converted to lowercase characters if the function is successful.

Example

```
if ( strToLower( "HELLO" ) == "hello" )
{
  ...
}
```

strToUpper

This function converts a string to uppercase characters.

Syntax

```
String strToUpper(String string);
```

Parameter

string
The string to convert to uppercase characters.

Return Values

Returns the string converted to uppercase characters if the function is successful.

Example

```
if ( strToUpper( "Hello" ) == "HELLO" )
{
  ...
}
```

Transaction Management Functions

Table 85-15 contains transaction management functions.

Table 85-15 Transaction Management Functions

Function	Description
edrDemandCancel	Sends a request to the Transaction Manager to cancel the current transaction.
edrDemandRollback	Sends a request to the Transaction Manager to roll back the current transaction.
edrRollbackReason	Allows the iScript module to request the reason for the rollback in the onRollback function.
tamItemType	Returns the type of an item in the currently processed transaction.
tamNumTransItems	Returns the number of items processed in the currently processed transaction.
tamStreamExtension	Used to access the extension value of each item in the current transaction.
tamStreamName	Used to access the stream name of each item in the current transaction.
tamTransId	Returns the transaction ID of the transaction currently being processed.

edrDemandCancel

This function sends a request to the Transaction Manager to cancel the current transaction.

Syntax

```
Bool edrDemandCancel();
```

Parameters

None.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

```
if ( edrDemandCancel() == false )
{
  logStdout( "ERROR: failed to demand cancel" );
}
```

edrDemandRollback

This function sends a request to the Transaction Manager to roll back the current transaction.

Syntax

```
Bool edrDemandRollback([rollbackReason]);
```

Parameter

rollbackReason
The reason for the rollback.

Return Values

Returns **true** if the function is successful. Returns **false** if the function fails.

Example

Request for rollback success status:

```
if ( edrDemandRollback() == false )
{
  logStdout( "ERROR: failed to demand rollback" );
}
```

Request for rollback with a reason:

```
edrDemandRollback("Invalid Input file")
```

edrRollbackReason

This function allows the iScript module to request the reason for the rollback in the onRollback function.

Syntax

```
String edrRollbackReason();
```

Parameters

None.

Return Values

Returns a string indicating the reason for the rollback.

Example

```
function Bool onRollback
{
  rollbackReason = edrRollbackReason();
  logStdout( "rollback reason= " + rollbackReason + "\n");
  return true;
}
```

tamItemType

This function returns the type of an item in the currently processed transaction. These items are only accessible for the functions dealing with transactions like onCancel, onCommit, onRollback, and so forth.

Syntax

```
Long tamItemType(Long idx);
```

Parameter

idx
The index of the transaction item you want to access.

Return Values

Returns the type of the specified item:

- TAM_NORMAL
- TAM_RECYCLE
- TAM_RECYCLE_TEST

Returns a value of <0 if there is no current transaction in all other functions or the index is out of range.

Example

```
function onCancel
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    if ( tamItemType(i) == TAM_NORMAL )
    {
      ...
    }
  }
}
```

tamNumTransItems

This function returns the number of items processed in the currently processed transaction. The count includes only items accessible for the functions dealing with transactions like onCancel, onCommit, onRollback, and so forth.

Syntax

```
Long tamNumTransItems();
```

Parameters

None.

Return Values

Returns the number of items in the transaction currently being processed. Returns **0** if there is no current transaction in all other functions or there are no items in the current transaction.

Example

```
function onCancel
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    ...
  }
}
```

tamStreamExtension

This function accesses the extension value of each item in the current transaction. The index should be between 0 and **tamNumTransItems()–1**. Usually, the extension value contains the sequence number of the currently processed stream.

Syntax

```
String tamStreamExtension(Long idx);
```

Parameter

idx
The index of the transaction item you want to access.

Return Values

Returns the stream extension string if the function is successful. Returns an empty string if the function fails.

Example

```
function onCommit
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    logFormat( "commiting " + tamStreamName(i) + \
    " with extension " + tamStreamExtension(i) );
  }
}
```

tamStreamName

This function accesses the stream name of each item in the current transaction. The index should be between 0 and **tamNumTransItems( )–1**.

Syntax

```
String tamStreamName(Long idx);
```

Parameter

idx
The index of the transaction item you want to access.

Return Values

Returns the stream name if the function is successful. Returns an empty string if the function fails.

Example

```
function onCommit
{
  Long i;
  for ( i=0; i<tamNumTransItems(); i=i+1 )
  {
    logFormat( "committing " + tamStreamName(i) );
  }
}
```

tamTransId

This function returns the transaction ID of the transaction currently being processed. This function should only be used with functions dealing with transactions like onCancel, onCommit, onRollback, and so forth.

Syntax

```
Decimal tamTransId();
```

Parameters

None.

Return Values

Returns the current transaction ID. Returns **0.0** if there is no current transaction in the other functions.

Example

```
function onCancel
{
  Decimal transId = tamTransId();
  ...
}
```

86
Pipeline Manager Input and Output Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager input and output modules.

EXT_InEasyDB

The EXT_InEasyDB module handles pipeline input from a database. See:

- About Getting Pipeline Input from a Database
- Configuring EDR Input Processing

This module runs automatically when you start Pipeline Manager.

Configure this module as a submodule of the INP_GenericStream module. See "INP_GenericStream".

To configure input from files, see "EXT_InFileManager".

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 86-1 lists the EXT_InEasyDB registry entries.

Table 86-1 EXT_InEasyDB Registry Entries

Entry	Description	Mandatory
ControlPath	Specifies the path for SQL, parameter, job and restart files. **./database/Oracle/Scripts/Suspense**	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
FieldDelimiter	Specifies the character that separates the EDR fields.	Yes
FileName	Specifies the job and restart file name prefix.	Yes
InputDirEmptyTime	Specifies the time period (in seconds), the input directory must be empty before the EVT_INPUT_DIR_EMPTY event is sent.	No
InputPrefix	Specifies the prefix of the stream/output file name.	No
InputSuffix	Specifies the flag which tells the stream to generate date and time information in the stream/output file name.	Yes
NumberOfRows	Specifies the array fetch size.	Yes
ParameterFile	Specifies the name of a file that contains iRule parameter values which can be used as placeholders in the SQL statements. They can be used in the update or startup registry.	Yes
Replace	Specifies the prefix/suffix should be replaced (**True**) or appended (**False**).	No

Table 86-1 (Cont.) EXT_InEasyDB Registry Entries

Entry	Description	Mandatory
SqlDetail	Specifies the name of a file that contains the SQL select statement for generating an EDR detail record. The choices are: • **StdRecycleDetail.sql** - used with the standard recycling feature. Used without changes. • **RecycleDetail.sql** - used with the Suspense Manager service integration component. You need to customize this file for your implementation. For details, see "Installing and Configuring Suspense Manager".	Yes
SqlHeader	Specifies the name of a file that contains the SQL select statement for generating an EDR header (result must be exactly one row).	No
SqlOnFailure	Specifies the name of a file that contains the SQL statement which is run if the output file is incorrect.	Yes
SqlOnSucess	Specifies the name of a file that contains the SQL statement that is run if the output file is correct.	Yes
SqlTrailer	Specifies the name of a file that contains the SQL select statement for generating an EDR trailer (result must be exactly one row).	No

Sample Registry

```
Stream
{
   ControlPath = ./database/Oracle/Scripts/Suspense
   DataConnection = IntegRate.DataPool.Login
   FileName = DB
   FileNameExtension = true
  inputPrefix = sol42_
  inputSuffix = .dat
   FieldDelimiter = ;
   ParameterFile = parameter.isc
   SqlHeader = header.sql
   SqlDetail = detail.sql
   SqlTrailer = trailer.sql
   SqlOnSuccess = success.sql
   SqlOnFailure = failure.sql
   Replace = true
   SynchronizeWithOutput = true
   NumberOfRows = 1000
}
```

Event Messages

Table 86-2 lists the EXT_InEasyDB event messages.

Table 86-2 EXT_InEasyDB Event Messages

Message	Description	Send/Recv
MSG_STREAM_START	The database input stream is started.	Send: Input module Send: Format

Table 86-2 (Cont.) EXT_InEasyDB Event Messages

Message	Description	Send/Recv
MSG_STREAM_END	The database input stream is stopped.	Send: Input module Send: Format
MSG_STREAM_BEGIN	The database stream starts the processing of a new input file.	Send: Input module
MSG_STREAM_END	The current file has been completely processed.	Send: Input module
MSG_STREAM_STOP	The database input stream is stopped (inactive).	Send: Input module
CMD_RENAME_INPUT_STREAM	The input file is renamed.	Receive: Output module

Events

Table 86-3 lists the EXT_InEasyDB events.

Table 86-3 EXT_InEasyDB Events

Event	Trigger	Parameter
EVT_CURSOR_OPENED	Starts the processing of a restart/ job file.	File name
EVT_INPUT_DIR_EMPTY	Control directory is empty.	Name of the control directory

EXT_InFileManager

The EXT_InFileManager module performs file handling for pipeline input from files. See:

- About Getting Pipeline Input from Files
- Configuring EDR Input Processing

This module runs automatically when you start Pipeline Manager.

Configure this module as a submodule of the INP_GenericStream module. See "INP_GenericStream".

To configure input from a database, see "EXT_InEasyDB".

Registry Entries

Table 86-4 lists the EXT_InFileManager registry entries.

Table 86-4 EXT_InFileManager Registry Entries

Entry	Description	Mandatory
DonePath	Specifies the path for the processed files.	Yes
DonePrefix	Specifies the prefix for the processed files.	No
DoneSuffix	Specifies the suffix for the processed files.	No
ErrorPath	Specifies the path for incorrect files.	Yes

Table 86-4 (Cont.) EXT_InFileManager Registry Entries

Entry	Description	Mandatory
ErrorPrefix	Specifies the prefix for incorrect files.	No
ErrorSuffix	Specifies the suffix for incorrect files.	No
InputDirEmptyTimeout	Specifies the time period (in seconds), the input directory must be empty before the EVT_INPUT_DIR_EMPTY event is sent.	No
InputPath	Specifies the path for the input files.	Yes
InputPrefix	Specifies the prefix for the input files.	No
InputSuffix	Specifies the suffix for the input files.	No
Replace	Specifies the prefix and or suffix can be replaced or appended. Default = **True**.	No
TempPrefix	Specifies the prefix for temporary files.	No

Sample Registry Section

```
InputStream
  {
    ModuleName = EXT_InFileManager
    Module
    {
      InputDirEmptyTimeout = 10
      InputPath   = ./samples/wireless/data/in
      InputPrefix = test
      InputSuffix = .edr
      DonePath    = ./samples/wireless/data/done
      DonePrefix  = test
      DoneSuffix  = .done
      ErrorPath   = ./samples/wireless/data/err
      ErrorPrefix = test
      ErrorSuffix = .err
      TempPrefix  = tmp
      Replace     = TRUE
    }
  }
```

Event Messages

Table 86-5 lists the EXT_InFileManager event messages.

Table 86-5 EXT_InFileManager Event Messages

Message	Description	Send/Receive
REQ_INPUT_FILENAME	Request to send back the complete input file name (including path) corresponding to a specific stream name (as known by the TAM).	Receive
REQ_INPUT_TEMP_FILENAME	Request to send back the temporary input file name (including path) corresponding to a specific stream name (as known by the TAM).	Receive

Table 86-5 (Cont.) EXT_InFileManager Event Messages

Message	Description	Send/Receive
REQ_DONE_FILENAME	Request to send back the final input file name (including path) corresponding to a specific stream name (as known by the TAM), after successful processing.	Receive
REQ_ERROR_FILENAME	Request to send back the final input file name (including path) corresponding to a specific stream name (as known by the TAM), after an unsuccessful processing.	Receive
REQ_RETURN_FILENAME	Request to send back the return file name (including path) corresponding to a specific stream name (as known by the TAM), when batch reject was requested.	Receive
REQ_RETURN_TEMP_FILENAME	Request to send back the temporary return file name (including path) corresponding to a specific stream name (as known by the TAM), when batch reject was requested.	Receive

EXT_OutFileManager

The EXT_OutFileManager module handles files for the OUT_Generic_Stream and OUT_Reject modules. See:

- Sending Output to a File
- Configuring EDR Output Processing

This module runs automatically when you start Pipeline Manager.

Registry Entries

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

Table 86-6 lists the EXT_OutFileManager registry entries.

Table 86-6 EXT_OutFileManager Registry Entries

Entry	Description	Mandatory
AppendSequenceNumber	Specifies if the sequence number should be appended to the output file name or not. See "Applying a Prefix to the Sequence Number".	No
DeleteEmptyFile	Deletes the output file if only the header and trailer are written to the stream. **Note:** By default, this entry is set to **True**. Configure any processes that manipulate output files to wait approximately one minute before acting on a file. This delay allows the module to delete empty files.	No

Table 86-6 (Cont.) EXT_OutFileManager Registry Entries

Entry	Description	Mandatory
Replace	Specifies if the prefix/suffix should be replaced (**TRUE**) or appended (**FALSE**). Default = **True**.	No
SequencerPrefix	This entry is used to specify a prefix to the sequence number before it gets appended to the generated output file name. This entry is used only when **AppendSequencerNumber** is set to **True**. See "Applying a Prefix to the Sequence Number".	No
TempPrefix	Specifies the prefix for the output stream's temporary data file. See "Configuring the Temporary File Name". Default = **.**	No
TempDataPath	Specifies the path for the internal temporary file list. **Important:** Do not change this registry entry. It is used by the pipeline for internal data processing.	No
TempDataPrefix	Specifies the prefix for the internal temporary file list. **Important:** Do not change this registry entry. It is used by the pipeline for internal data processing.	No
TempDataSuffix	Specifies the suffix for the internal temporary file list. **Important:** Do not change this registry entry. It is used by the pipeline for internal data processing.	No
OutputPath	Specifies the path for the output files. See "Configuring File Prefixes and Suffixes".	Yes
OutputPrefix	Specifies the prefix for the output files. See "Configuring File Prefixes and Suffixes".	No
OutputSuffix	Specifies the suffix for the output files. See "Configuring File Prefixes and Suffixes".	No
UseInputStreamName	Specifies to use the input file name to build the output file name. See "Creating an Output File Name from the Input File Name".	No

Sample Registry

```
#------------------------------------------------------------------
# The /service/telco/gsm/telephony output stream
#------------------------------------------------------------------
TELOutput
{
  ModuleName = OUT_GenericStream
  Module
  {
    Grammar = ./formatDesc/Formats/Solution42/SOL42_V430_REL_OutGrammar.dsc
    DeleteEmptyStream = True
    OutputStream
      {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = ./samples/wireless/data/telout
        OutputPrefix = test
```

```
        OutputSuffix = .out
        UseInputStreamName = [2,4;4,6;8,&]
        TempPrefix   = tmp.

        TempDataPath    = ./samples/wireless/data/telout
        TempDataPrefix  = tel.tmp.
        TempDataSuffix  = .data
        Replace = TRUE
        SequencerPrefix = "+"
        }
    }
  }
} # end of TELOutput
```

Messages and Requests

Table 86-7 lists the EXT_OutFileManager messages and requests.

Table 86-7 EXT_OutFileManager Messages and Requests

Message	Description	Send/Receive
REQ_EVENTHANDLER_NAME	Get the event handler.	Send

Events

Table 86-8 lists the EXT_OutFileManager events.

Table 86-8 EXT_OutFileManager Events

Event	Trigger	Parameter
EVT_OUTPUT_FILE_READY	The renaming from the temporary file to the output file.	Target file name

INP_GenericStream

The INP_GenericStream module provides the input interface to pipelines. See "Configuring EDR Input Processing".

Registry Entries

Table 86-9 lists the INP_GenericStream registry entries.

Table 86-9 INP_GenericStream Registry Entries

Entry	Description	Mandatory
DefaultOutput	The default output stream.	
Grammar	Path to the input grammar description file.	No
InputStream	The input submodule: • EXT_InEasyDB • EXT_InFileManager	

Sample Registry for INP_GenericStream

```
InputModule
{
  ModuleName  = INP_GenericStream
  Module
  {
    DefaultOutput = EdrOutput
    Grammar       = ./FMD/Formats/Solution42/SOL42_V430_InGrammar.dsc
    InputStream
      ModuleName = EXT_InFileManager
      Module
      {
        InputPath   = ../input/maxitel/in
        InputPrefix = Sol42
        InputSuffix = .edr
        DonePath    = ../input/maxitel/done
        DonePrefix  = Sol42
        DoneSuffix  = .done
        ErrorPath   = ../input/maxitel/err
        ErrorPrefix = Sol42
        ErrorSuffix = .err
        TempPrefix  = tmp
        Replace     = TRUE
      }
    }
  }
}
```

INP_Realtime

The INP_Realtime module converts data in flist format to the EDR container format. See "Configuring a Real-Time Discounting Pipeline".

You can use an iScript to overwrite the mappings from flist fields to EDR fields, and to add custom mappings.

For more information, see "About Customizing Mapping of Flist Fields to Rating EDR Container Fields".

Registry Entries

Table 86-10 lists the INP_Realtime registry entries.

Table 86-10 INP_Realtime Registry Entries

Entry	Description	Mandatory
DefaultOutput	Specifies the default output module. This is always the OUT_Realtime module.	Yes
MappingScript	iScript file name.	No

Table 86-10 (Cont.) INP_Realtime Registry Entries

Entry	Description	Mandatory
OpcodeMapping	Specifies the XML file that describes the input flist field to EDR container field mapping. BRM provides the following default flist-to-EDR mappings: • Discounting: **discount_event.xml** • Rerating: **rate_event.xml** • Zoning: **zonemap_event.xml** You can customize these files to change how flists are mapped to EDR container fields. See "About Customizing Mapping of Flist Fields to Rating EDR Container Fields".	Yes

Sample Registry Entry

In this example, **rate_event.xml** maps rerate requests in flist format to EDR container fields.

```
InputModule
{
  ModuleName = INP_Realtime
  Module
  {
     DefaultOutput = PcmOutput
     OpcodeMapping = ./formatDesc/Formats/Realtime/rate_event.xml
     MappingScript = ./inflist.isc
  }
```

INP_Recycle

The INP_Recycle module is used by standard recycling and Suspense Manager in the pre-recycling pipeline. It reads suspended usage records from the BRM database, restores original EDRs, applies edits to them, and pushes EDRs into the pre-recycling pipeline.

- For standard recycling, see "Configuring a Pre-Recycling Pipeline".
- For Suspense Manager, see "Configuring a Pre-Recycling Pipeline".

Dependencies

Requires connections to:

- DAT_Recycling module
- BRM database

Registry Entries

Table 86-11 lists the INP_Recycle registry entries.

Table 86-11 INP_Recycle Registry Entries

Entry	Description	Mandatory
DefaultOutput	Specifies the default output stream.	Yes

Table 86-11 (Cont.) INP_Recycle Registry Entries

Entry	Description	Mandatory
InfranetConnection	Specifies a connection to the BRM database.	Yes
InputStream	Configures the EXT_InEasyDB module. See "EXT_InEasyDB".	Yes
RecyclingDataModule	Specifies a connection to the DAT_Recycling module.	Yes

Sample Registry

```
InputModule
{
 ModuleName            = INP_Recycle
  RecyclingDataModule   = ifw.DataPool.RecyclingData
  Module
  {
    DefaultOutput = TELOutput
    RecyclingDataModule = ifw.DataPool.RecyclingData
    InfranetConnection = ifw.DataPool.LoginInfranet
    InputStream
    {
      ModuleName = EXT_InEasyDB
      Module
      {
        ControlPath           = ./data/input/db
        DataConnection        = ifw.DataPool.LoginInfranet
        FileName              = DB
        FileNameExtension     = true
        InputPrefix           = sol42_
        InputSuffix           = .dat
        FieldDelimiter        = \t
        RecordDelimiter       = \n
        ParameterFile         = parameter.isc
        # optional parameter:
        SqlHeader             = RecycleHeader.sql
        SqlDetail             = StdRecycleDetail.sql
        # optional parameter:
        # SqlTrailer            = trailer.sql
        SqlOnSuccess          = success.sql
        SqlOnFailure          = failure.sql
        Replace               = true
        SynchronizeWithOutput = true
        NumberOfRows          = 1000
      }
    } # end of InputStream
  }
} # end of InputModule
```

EDR Container Fields

Table 86-12 lists the INP_Recycle EDR container fields.

Table 86-12 INP_Recycle EDR Container Fields

Alias field name Default field name	Type	Access	Description
OVERRIDE_REASONS DETAIL.ASS_SUSPENSE_OVERRIDE_REASONS	String	Write	The suspense reasons to ignore during recycling. Used by the other pipeline modules for rating call records in spite of the pipeline validation rules they violate.
PIPELINE_NAME DETAIL.ASS_SUSPENSE_EXT.PIPELINE_NAME	String	Write	Name of the pipeline originally used for the EDR. Read from the database. Used by the IRL_PipelineSplitting module.
SUSPENSE_ID DETAIL.ASS_SUSPENSE_EXT.SUSPENSE_ID	Integer	Write	The POID ID of the */suspended_usage* object for this EDR. Used by Suspended Event Loader when updating suspended usage records.
BATCH_ID DETAIL.ORIGINAL_BATCH_ID	String	Write	The original routing switch batch ID. Used for revenue assurance.
PROCESS_STATUS DETAIL.INTERN_PROCESS_STATUS	Integer	Write	Indicates whether the EDR is being recycled (**1**) or test recycled (**2**).

INP_Restore

The INP_Restore module reads serialized EDRs from the file output by the OUT_Serialize module. It restores EDRs to normal format and pushes them into a pipeline.

Dependencies

Requires the use of the OUT_Serialize module to produce serialized EDRs in the correct format.

When you configure this module, you also configure the EXT_InFileManager module, which manages input, temporary, and done files. See "EXT_InFileManager" in the Infranet documentation for more information about this module.

Registry Entries

Table 86-13 lists the INP_Restore registry entries.

Table 86-13 INP_Restore Registry Entries

Entry	Description	Mandatory
DefaultOutput	Specifies the default output stream.	Yes
InputStream	Configures the EXT_InFileManager module.	Yes

Sample Registry

```
#--------------------------------
# Input section
#--------------------------------
Input
{
  UnitsPerTransaction             = 1

  InputModule
  {
    ModuleName                    = INP_Restore
    Module
    {
      DefaultOutput               = EdrOutput

      InputStream
      {
        ModuleName                = EXT_InFileManager
        Module
        {
          InputPath               = $DATA/in
          InputPrefix             = testpipeline
          InputSuffix             = .edr

          DonePath                = $DATA/done
          DonePrefix              = testpipeline
          DoneSuffix              = .done

          ErrorPath               = $DATA/err
          ErrorPrefix             = testpipeline
          ErrorSuffix             = .err

          TempPrefix              = tmp

          Replace                 = True

          InputDirEmptyTimeout    = 60
        }
      }
    }
  }
}
```

OUT_DB

The OUT_DB module sends output to the Pipeline Manager database.

See "Sending Output to a Database".

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 86-14 lists the OUT_DB registry entries.

Table 86-14 OUT_DB Registry Entries

Entry	Description	Mandatory
ControlPath	Specifies the name of the control/configuration directory. See "About the OUT_DB Module Configuration Files".	Yes
DataConnection	Specifies the connection to the Pipeline Manager database.	Yes
DeleteWithoutDetails	Specifies if an empty output stream should force a rollback of all actions. Default = **True** See "Handling Empty Output Streams ".	No
Destination	Specifies the value for the destination field in the header record. See "Specifying the Destination ".	No
DetailTableDefinition	Specifies the name of the file that contains the description of the destination detail table. See "About the OUT_DB Module Configuration Files".	Yes
FieldDelimiter	Specifies the delimiter between each field needed by the tokenizer. See "About the OUT_DB Module Configuration Files".	Yes
HeaderTableDefinition	Specifies the name of the file that contains the description of the destination header table. See "About the OUT_DB Module Configuration Files".	No
NumberOfRows	Specifies the array size for the bulk inserter. A good value is **500**. See "About the OUT_DB Module Configuration Files".	Yes
ParameterFile	Specifies the name of parameter file which contains optional key/value entries See "Parameter File".	Yes
RowNumAlias	Specifies the alias that is replaced by the inserted rows. See "About the OUT_DB Module Configuration Files".	Yes
SaveConfigurationFile	Specifies whether to delete the configuration file of each stream. Default = **False** See "About the OUT_DB Module Configuration Files".	No
Source	Specifies the value for the source field in the header record. See "Specifying the Source ".	No
SqlBeginStream	Specifies the name of SQL file that contains an SQL statement. See "SqlBeginStream ".	No
SqlEndStream	Specifies the name of SQL file that contains an SQL statement. See "SqlEndStream ".	No
StreamNameAlias	Specifies the alias that is replaced by the internal stream name value. See "About the OUT_DB Module Configuration Files".	No
TrailerTableDefinition	Specifies the name of the file that contains the description of the destination trailer table. See "About the OUT_DB Module Configuration Files".	No
WriteDefaultEdr	Specifies if a default EDR is written in empty data streams. Default = **False** See "Handling Empty Output Streams ".	No

Sample Registry

```
Streams
{
  EdrOutput
  {
    StreamDestination       = Database
    Destination             = CBC21
    Source                  = D1
    DataConnection          = integrate.DataPool.Login
    NumberOfRows            = 500
    ControlPath             = control
    FieldDelimiter          = ;
    RowNumAlias             = __RowNum__
    StreamNameAlias         = __StreamName__
    SqlBeginStream          = beginStream.sql
    SqlEndStream            = endStreamNeu.sql
    ParameterFile           = parameter_out.isc
    HeaderTableDefinition   = headerTable.dat
    DetailTableDefinition   = detailTable.dat
    TrailerTableDefinition  = trailerTable.dat
    WriteDefaultEdr         = false
    DeleteWithoutDetails    = true
    SaveConfigurationFile   = true
  }
  Reject
  {
    StreamDestination       = File
    OutputPath              = /data/reject
    OutputSuffix            = .rej
    Replace                 = True
  }
}
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

OUT_DevNull

The OUT_DevNull module removes EDRs that are not needed by Pipeline Manager. See "Configuring Output of Discarded EDRs".

For more information, see "Discarding and Skipping EDRs ".

Sample Registry

```
OutputCollection
    {
    DevNull
    {
        ModuleName = OUT_DevNull
        Module
        {
```

```
        }
    }
```

OUT_GenericStream

The OUT_GenericStream module handles the output stream for rated EDRs. See "Configuring EDR Output Processing".

When you configure the OUT_GenericStream module, you configure the EXT_OutFileManager module to specify file management options. See "EXT_OutFileManager".

Registry Entries

Table 86-15 lists the OUT_GenericStream registry entries.

Table 86-15 OUT_GenericStream Registry Entries

Entry	Description	Mandatory
AddInvoiceData	When set to True, the output module adds invoice data to each BRM billing record. Default = **False** See "Adding Pipeline Rating Data to an Invoice".	No
DeleteEmptyStream	Specifies to delete empty output streams. If you run multiple RE Loader processes in parallel, set this option to **True**. Otherwise, RE Loader attempts to load the empty files. Default = **True** See "Configuring Pipeline Manager to Delete Empty Output Streams".	No
EventType	Specifies the BRM event type that the output file contains, such as **/event/delayed/session/gsm**.	Mandatory only for RE Loader-related pipelines
Grammar	Specifies the output grammar description file.	Yes
OutputStream	Contains the configuration for the EXT_OutFileManager.	Yes
ProcessType	Specifies which process created the output file and can be set to one of the following values: • **RATING_PIPELINE** • **EVENT_EXTRACTION_TOOL** • **BACKOUT_PIPELINE** • **RERATING_PIPELINE** • **PIN_TRANSFORM_CDR**	Mandatory only for RE Loader-related pipelines
Sequencer	Specifies the Sequencer for performing sequence generation. This Sequencer must be defined in the **SequencerPool** section of the registry file.	No

Sample Registry

```
#------------------------------------------------------------------
# The /service/telco/gsm/telephony output stream
#------------------------------------------------------------------
TELOutput
{
  ModuleName = OUT_GenericStream
  ProcessType = RATING_PIPELINE
  EventType = /event/delayed/session/gsm
```

```
  Module
  {
    Grammar = ./formatDesc/Formats/Solution42/SOL42_V430_REL_OutGrammar.dsc
    DeleteEmptyStream = True
    Sequencer = SequenceGenerator_1
    OutputStream
      {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = ./samples/wireless/data/telout
        OutputPrefix = test
        OutputSuffix = .out
        TempPrefix   = tmp.
        TempDataPath    = ./samples/wireless/data/telout
        TempDataPrefix  = tel.tmp.
        TempDataSuffix  = .data
        Replace = TRUE
        }
     }
  }
} # end of TELOutput
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

OUT_Realtime

The OUT_Realtime module converts data in the pipeline EDR output to flist format. It sends the output to the NET_EM module automatically. You don't need to configure a pointer to the NET_EM module.

You can use an iScript to overwrite the mappings and to add custom mappings. You use the registry file to specify the iScript.

For more information, see "Configuring a Real-Time Discounting Pipeline".

Registry Entries

Table 86-16 lists the OUT_Realtime registry entries.

Table 86-16 OUT_Realtime Registry Entries

Entry	Description	Mandatory
MappingScript	iScript file name.	No
DoRating	Specifies the pipeline to be used. Set to **False** for Discounting/Zoning pipeline. Set to **True** for Rerating pipeline.	Yes

Sample Registry Entry

```
OutputCollection
{
  PcmOutput
  {
    ModuleName = OUT_Realtime
    Module
    {
       DoRating = false
       #MappingScript = <Optional Output IScript>
    }
  }
}
```

OUT_Reject

The OUT_Reject module writes rejected EDRs to an output stream. The written record is exactly the same as the original input record. See "Configuring Output for Rejected or Duplicate EDRs".

When you configure the OUT_Reject module, you configure the EXT_OutFileManager module to specify file management options. See "EXT_OutFileManager".

Sample Registry

```
Reject
{
  ModuleName = OUT_Reject
  Module
  {
    OutputStream
    {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = ../output/tel/rej
        OutputPrefix = Sol42
        OutputSuffix = rej
        TempPrefix   = tmp
        Replace = TRUE
        DeleteEmptyFile = FALSE
      }
    }
  }
}
```

> **Note:**
>
> To ensure output file integrity, specify a unique combination of OutputPath, OutputSuffix, and OutputPrefix values for each output stream defined in the registry.

OUT_Serialize

The OUT_Serialize module creates serialized EDR records with base64 encoding.

Dependencies

The INP_Restore module is required to read and restore EDRs serialized by this module.

When you configure this module, you also configure the EXT_OutFileManager module, which manages output files. See "EXT_OutFileManager" in the Infranet documentation for more information about this module.

Registry Entries

Table 86-17 lists the OUT_Serialize registry entries.

Table 86-17 OUT_Serialize Registry Entries

Entry	Description	Mandatory
DeleteEmptyStream	Specifies whether empty streams should be deleted. Set to **True** to prevent SEL from attempting to load empty files. Default = **True**	Yes
ProcessType	Specifies which process created the output file. If used, sets the HEADER.CREATION_PROCESS field in the EDR to the value specified.	No
EventType	Specifies the Infranet event type that the output file contains, such as **/event/delayed/session/gprs**. If used, sets the HEADER.EVENT_TYPE field to the values specified.	No
OutputStream	Configures the EXT_OutFileManager.	Yes

Sample Registry

```
#------------------------------------------------------------
# The serialized EDR output stream
#------------------------------------------------------------
SerEdrOutput
{
  ModuleName = OUT_Serialize
  Module
  {
    DeleteEmptyStream = True
    ProcessType = RECYCLING_PIPELINE
    EventType = /event/delayed/session/gprs

    OutputStream
    {
      ModuleName = EXT_OutFileManager
      Module
      {
        OutputPath = $DATA/out
        OutputPrefix = test
        OutputSuffix = .out
```

```
        TempPrefix    = .
        TempDataPath     = $DATA/out
        TempDataPrefix   = tel.tmp.
        TempDataSuffix   = .data

        Replace = TRUE
      }
    }
  }
} # end of SerEdrOutput
```

Pipeline Dispatcher

Parses CDR files to multiple identical pipelines. This module routes files with a specified filename prefix and suffix from a single input directory to multiple pipelines in round-robin fashion.

Registry Entries

Table 86-18 lists the Pipeline Dispatcher registry entries.

Table 86-18 Pipeline Dispatcher Registry Entries

Entry	Value	Description	Mandatory
DispatcherName	N/A	Specifies the name of the dispatcher. You can use any name, for example, **Dispatcher1**. If your system requires multiple dispatchers, create a set of entries for each dispatcher.	N/A
DispatcherName.**InputPath**	String	Specifies the path of the CDR input directory. For example: **InputPath = ./samples/wireless/data/input**	N/A
DispatcherName.**InputPrefix**	String	Specifies the prefix of all CDR files you want routed. The dispatcher only routes files with the specified prefix and suffix to your pipelines. For example: **InputPrefix = test**	N/A
DispatcherName.**InputSuffix**	String	Specifies the suffix of all CDR files you want routed. The dispatcher only routes files with the specified prefix and suffix to your pipelines. For example: **InputSuffix = .edr**	N/A
DispatcherName.**TargetPipelines**	N/A	Subgroup that lists which pipelines to route your CDR files.	N/A
DispatcherName.**TargetPipelines.DestinationPipeline**	String	Specifies to which pipelines to route CDR files. Add an entry for each pipeline. For example: **DestinationPipeline = ifw.Pipelines.W_SAMPLE** **DestinationPipeline = ifw.Pipelines.W_SAMPLE_2** **Important:** The pipelines must use a separate input directory from the CDR input files.	N/A

Table 86-18 (Cont.) Pipeline Dispatcher Registry Entries

Entry	Value	Description	Mandatory
DispatcherName.**TargetPipelines.Routing**	ROUND_ROBIN	Specifies to use round-robin routing. **Note:** The dispatcher uses round-robin routing only.	Yes

Sample Registry

```
ifw
{
    PipelineDispatcher
    {
        ModuleName = EXT_PipelineDispatcher
        Module
        {
            Dispatcher1
            {
                InputPath = ./samples/wireless/input
                InputPrefix = TAP3
                InputSuffix = .edr

                TargetPipelines
                {
                    DestinationPipeline = ifw.Pipelines.Pipeline_1
                    DestinationPipeline = ifw.Pipelines.Pipeline_2
                    Routing = ROUND_ROBIN
                }
            }
        }
    }
}
```

87

Pipeline Manager Framework Modules

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager framework modules.

Controller

Use the Pipeline Manager Controller to control and monitor components in the Pipeline Manager framework. The Controller also generates the log message table that is used by the LOG module to generate the Process log file, the Pipeline Manager log file, and the Stream log file. For information, see "LOG".

Registry Entries

Table 87-1 lists the Controller registry entries.

Table 87-1 Controller Registry Entries

Entry	Value	Description	Mandatory?
Active	**True** **False**	Activates or deactivates the Pipeline Manager framework. • **True** activates the Pipeline Manager framework. • **False** deactivates the Pipeline Manager framework.	Yes
DataPool	N/A	Subgroup that contains entries for all data modules in the Pipeline Manager framework.	Yes
DiagnosticDataHandler	N/A	Subgroup that configures diagnostic data collection.	No
EventHandler	N/A	Subgroup that contains the Event Handler entries.	No
Instrumentation	N/A	Subgroup that configures Operations Management Framework (OMF) instrumentation data collection.	Yes
Instrumentation.ProbeBroker	String	Specifies the path to the OMF instrumentation folder.	Yes
Instrumentation.HttpServer	String	Specifies configuration data for the OMF HTTP server.	No
LogMessageTable.MessageFilePath	String	Specifies the path to your error message files. By default, the Pipeline Manager installer installs your error message files in the *pipeline_home***/etc** directory. *pipeline_home* is the directory where you installed Pipeline Manager.	No
LogMessageTable.MessageFilePrefix	String	Specifies the prefix for your error message files.	No
LogMessageTable.MessageFileSuffix	String	Specifies the suffix for your error message files.	No
MemoryMonitor	N/A	Subgroup that configures memory monitoring. See "Memory Monitor".	No
Pipelines	N/A	Subgroup that contains the entries for each pipeline.	Yes

Table 87-1 (Cont.) Controller Registry Entries

Entry	Value	Description	Mandatory?
ProcessLog	N/A	Subgroup that contains entries for the main processing log.	Yes
ProcessLoopTimeout	Integer	Specifies the interval, in seconds, between polling for a new semaphore file. This parameter controls the overall event loop, which includes looking for semaphore files. The value must be greater than **0**.	Yes
QueueRequestTimeout	Integer	Specifies the interval for logging buffer sizes, in seconds. A value of **0** turns off logging.	Yes
Registry	N/A	Subgroup that contains registry processing entries.	Yes
Registry.Buffer	N/A	Subgroup that specifies the registry buffer's size and type. The registry entries in this subgroup depend on the buffer type.	Yes
Registry.FileName	String	Specifies the name of a file in dot-separated format that contains current updated registry settings, including semaphore updates.	Yes
Registry.FilePath	String	Specifies the path to the registry file.	Yes
Registry.NiceFormatFileName	String	Specifies the name of a file that contains current updated registry settings, including semaphore updates. This file format is easier to read than **.FileName** format and can be used to debug a registry file or create a new one.	Yes
Semaphore	N/A	Subgroup that contains semaphore processing entries.	Yes
Semaphore.FileName	String	Specifies the name of the semaphore file.	Yes
Semaphore.FilePath	String	Specifies the path to the semaphore file.	Yes
Semaphore.RetainFiles	**True** **False**	Specifies whether semaphore files are deleted or saved after they are processed. • **True** specifies to save semaphore files. The Controller renames the file by appending the current timestamp to the file name in the format *YYYYMMDD_hhmmss* and logs the semaphore file's new name in the **process.log** file. For example, the **semaphore.reg** file is renamed **semaphore_20031022_120803.reg**. • **False** specifies to delete semaphore files immediately after they are processed. The default is **False**.	No
TransactionIDController	N/A	Subgroup that contains the Transaction ID Controller entries.	Yes

Sample Registry File

```
ifw
{
    Active = TRUE
    ProcessLoopTimeout = 10
    QueueRequestTimeout = 0
```

```
    Semaphore
    {
        FilePath = ./semaphore
        FileName = semaphore.reg
        RetainFiles = False
    }
    Registry
    {
        FilePath = ./info
        FileName = Sample.reg
        NiceFormatFileName = niceSample.reg
        Buffer
        {
            Size = 1000
        }
    }
    ProcessLog
    {
        ModuleName = LOG
        Module
        {
        }
    }
    LogMessageTable
    {
        MessageFilePath= ./etc
        MessageFileSuffix= .msg
    }
    EventHandler
    {
        ModuleName = EVT
        Module
        {
        }
    }
    DataPool
    {
        Login
        {
            ModuleName = DBC
            Module
            {
                UserName = TEST
                PassWord = password
                DatabaseName = TE01
                AccessLib = oci11g72
                Connections = 5
            }
        }
    }
    TransactionIdController
    {
        Source = Database
        Generator
        {
            DataConnection = integrate.DataPool.Login
        }
    }
    Pipelines
    {
    }
}
```

Semaphore File Entries

Table 87-2 lists the Controller Semaphore file entries.

Table 87-2 Controller Semaphore File Entries

Entry	Description
Active	Starts and stops the Pipeline Manager framework.
LogTimeStatistic	Specifies whether to write time statistics into the process log file.
QueueRequestTimeout	Specifies the interval for logging buffer sizes, in seconds. A value of **0** turns off logging.

Sample Semaphore Entry

```
ifw.Active = True
```

Events

Table 87-3 lists the Controller Events.

Table 87-3 Controller Events

Event	Trigger
EVT_INTEGRATE_START	Pipeline Manager starts processing.
EVT_INTEGRATE_STOP	Pipeline Manager terminates. No more files are processed.

Database Connect (DBC)

Connects the Pipeline Manager framework to the Pipeline Manager database.

Registry Entries

Table 87-4 lists the Database Connect registry entries.

Table 87-4 Database Connect Registry Entries

Entry	Value	Description	Mandatory?
AccessLib	**oci10g72** **oci11g72**	Specifies the name of the database access library, without the **lib** prefix and **.so** suffix. • Use **oci10g72** for Oracle10g databases. • Use **oci11g72** for Oracle11g databases.	Yes
Connections	Integer	Specifies the number of database connections that a database module holds open in a connection pool. The default is **1**.	No
DatabaseName	String	Specifies the database name.	Yes
PassWord	String	Specifies the encrypted database password in hexadecimal format.	Yes

Table 87-4 (Cont.) Database Connect Registry Entries

Entry	Value	Description	Mandatory?
ServerName	String	Specifies the server name. The default is **''**.	No
UserName	String	Specifies the database user name.	Yes
ConvertToUpperCase	True False	Specifies whether to convert the user name used by the password decryption to uppercase. The default is **True**.	No

Sample Registry for Oracle Databases

```
ifw
{
    DataPool
    {
    ...
        #------------------------------------------------------------
        # Database Connection Pipeline
        #------------------------------------------------------------
        Login
        {
            ModuleName = DBC
            Module
            {
                UserName = USER
                PassWord = 574B93D1CBF21D1161611E0A07
                DatabaseName = ORA_DB
                AccessLib = oci11g72
                Connections = 5
            }
        }
    ...
    }
}
```

Semaphore Entries

Table 87-5 lists the Database Connect Semaphore entry.

Table 87-5 Database Connect Semaphore Entry

Parameter	Description	Mandatory
Reconnect	Closes all open database connections and reconnects to the database.	No

Sample Semaphore

```
ifw.DataPool.Login.Module.Reconnect {}
```

EDR Factory

Use the EDR Factory to generate and allocate memory to EDR containers. The EDR Factory uses a container description file to generate each container.

Registry Entries

Table 87-6 lists the EDR Factory registry entries.

Table 87-6 EDR Factory Registry Entries

Entry	Value	Description	Mandatory?
DataConnection	String	Specifies the registry name of the database connection module (DBC). When you use this entry, the EDR Factory connects to the IFW_ALIAS_MAP database table to retrieve alias names. **Note:** Use this field when you use aliases to describe EDR container fields.	No
Description	String	Specifies the path to the container description file. See "About the Container Description File".	Yes
EdrVersionDataConnection	String	Specifies the registry name of the database connection module (DBC). When you use this entry, the EDR Factory connects to the EDR_FIELD_MAPPING_T database table to retrieve the EDR field mapping.	No
UsageStatistic	String	Specifies the name of the usage statistic file. This file lists all modules that are using EDR container fields together with the fields that are accessed by each module.	No

Sample Registry

```
EdrFactory
{
    DataConnection = integrate.DataPool.Login
    Description = ./FMD/Portal/containerDesc.dsc
    UsageStatistic = edrStatistic.txt
    EdrVersionDataConnection = ifw.DataPool.LoginInfranet
}
```

Event Handler

Use the Event Handler to start external programs when triggered by internal Pipeline Manager events.

Registry Entries

Table 87-7 lists the Event Handler registry entries.

Table 87-7 Event Handler Registry Entries

Entry	Description	Mandatory?
Buffer	Subgroup that specifies the size and type of the Event Handler's internal queue buffer. The registry entries in this subgroup depend on the buffer type.	Yes
Events	Subgroup that contains trigger entries.	Yes

Table 87-7 (Cont.) Event Handler Registry Entries

Entry	Description	Mandatory?
Event.*EventName*	Specifies the event that triggers an external program. Add an entry for each event that triggers an external program.	Yes
Event.*ModuleSendingEvent*	Specifies the registry name of the module that sends the event to the Event Handler. Add an entry for each module that can trigger an external program.	Yes
TimeToWait	Specifies the time in seconds that the Event Handler waits for the external program to terminate. By default, no **TimeToWait** value is assumed.	No

Sample Registry

```
EventHandler
{
    ModuleName = EVT
    Module
    {
        Events
        {
            ifw.DataPool.Customer.Module
            {
                EVT_ReloadSuccess = ./script/script_1
                EVT_ReloadFailed = ./script/script_2
                TimeToWait = 30
            }
            ifw.Pipelines.*
            {
                EVT_PIPELINE_START = ./script/script_3
                TimeToWait = 10
            }
        }
        Buffer
        {
            Size = 10
        }
    }
}
```

Instances

Use the Instances module to configure multiple instances of sequencers, output streams, or system brands for multiple roaming partners. This module is optional.

> **Note:**
>
> This module does not configure multiple instances of pipelines. To do that, use the **ifw.Pipelines.Instances** subsection.

Registry Entries

Table 87-8 lists the Instances registry entries.

Table 87-8 Instances Registry Entries

Entry	Description	Mandatory?
InstantiationName	Specifies the descriptive name of the instantiation, for example, TAPOutputStreamsInstantiation.	Yes, if the Instances module is used.
InstantiationName.**BlockName**	Specifies the template section or entry in the roaming registry file that is used to instantiate multiple registry sections or entries. The template section or entry contains variables for the section name, entry name, or the value of the entry that must be changed in each new instance created.	Yes
InstantiationName.**DataFile**	Specifies the path to the data file generated by the **RoamingConfigGen64** utility. See "RoamingConfigGen64" for more information.	Yes
InstantiationName.**InstanceSpecificEntries**	Subgroup that specifies the entries that must be changed in each new instance created, such as the section name, entry name, the value of an entry, the change mode, and so on.	Yes
InstantiationName.**InstanceSpecificEntries**.*InstanceChangeName*	Specifies the descriptive name of the change required in each instance; for example, ModifyBlockName.	Yes
InstantiationName.**InstanceSpecificEntries**.*InstanceChangeName*.**Instance**	Specifies whether to change the section name, entry name, or the value of the entry in each new instance created. Valid values are: • **[BlockName]** specifies that the section name or entry name must be changed in each new instance. For example, to change the section name of the **SequencerPool.SEQ_GEN_TAPOUT_XXX** subsection in each new instance (such as **SEQ_GEN_TAPOUT_OPR01**, **SEQ_GEN_TAPOUT_OPR02**, and so on), set the **Instance** entry to **[BlockName]**. • **[BlockValue]** specifies that the value of the entry must be changed in each new instance. **Note:** Use this value only if *InstantiationName*.**BlockName** is a template entry: do not use this value if it is a template section. For example, to change the value of the **SystemBrands.XXX** entry in each new instance (such as TAPOutput_OPR01, TAPOutput_OPR02, and so on), set the **Instance** entry to **[BlockValue]**. • *RegistryEntry* specifies the entry in the template registry section for which the value must be changed in each new instance. **Note:** Use this value only if *InstantiationName*.**BlockName** is a template section: do not use this value if it is a template entry. For example, to change the value of the **Module.Recipient** entry in the **TAPOutput_XXX** section, set the **Instance** entry to **Module.Recipient**.	Yes

Table 87-8 (Cont.) Instances Registry Entries

Entry	Description	Mandatory?
InstantiationName.**InstanceSpecificEntries**.*InstanceChangeName*.**UseColumn**	Specifies the column in the data file generated by the **RoamingConfigGen64** utility. This column is used to change the section name, entry name, or the value of the entry in each instance according to the change mode. For example, the TAPOUT_SEQUENCER column is used to change the section name in each instance of the **SequencerPool.SEQ_GEN_TAPOUT_XXX** subsection. See "Sample Registry for Multiple Instances of Sequencers".	Yes
InstantiationName.**InstanceSpecificEntries**.*InstanceChangeName*.**Mode**	Specifies the mode of changing the section name, entry name, or the value of the entry in each instance using the column values in the data file generated by the **RoamingConfigGen64** utility. Valid values are: • **REPLACE** specifies that the section name, entry name, or the value of the entry is replaced with the corresponding column value from the data file. For example, if the entry name is *XXX* and the corresponding column value is **OPR01**, *XXX* is replaced with **OPR01** in the newly created instance. • **PREFIX** specifies that the corresponding column value is prefixed with the section name, entry name, or the value of the entry in each instance. For example, if the value of the entry is **.tmp** and the corresponding column value is **OPR01**, **OPR01** is prefixed with **.tmp** and the value is added as **OPR01.tmp** in the newly created instance. • **SUFFIX** specifies that the corresponding column value is suffixed with the section name, entry name, or the value of the entry in each instance. For example, if the value of the entry is **NREUR01** and the corresponding column value is **OPR01**, **OPR01** is suffixed with **NREUR01** and the value is added as **NREUR01OPR01** in the newly created instance. The default mode is **REPLACE**.	No

Sample Registry for Multiple Instances of Sequencers

```
Instances
{
   SEQ_GEN_TAPOUT
    {
     BlockName  = SequencerPool.SEQ_GEN_TAPOUT_XXX
     DataFile = ./RoamingPartnerConf.dat
     InstanceSpecificEntries
     {
         ModifyBlockName
         {
           Instance = [BlockName]
           UseColumn = TAPOUT_SEQUENCER
          }
      }
    }
}
SequencerPool
{
SEQ_GEN_TAPOUT_XXX
{
  Source = Database
  Controller
```

```
  {
   SequencerType = Generation
   ReuseGap = True
   SequenceLength = 5
   DatabaseConnection = ifw.DataPool.Login
  }
}
```

Sample Registry for Multiple Instances of System Brands

```
Instances
{
 EventSplitting
  {
   BlockName  =
Pipelines.TAPOutCollectPipeline.Functions.Processing.FunctionPool.RoamPartner_EventSplitt
ing.Module.SystemBrands.XXX
   DataFile = ./RoamingPartnerConf.dat
   InstanceSpecificEntries
   {
     ModifyBlockName
     {
       Instance = [BlockName]
       UseColumn = VPLMN
     }
     ModifyBlockValue
     {
      Instance = [BlockValue]
      UseColumn = TAPOUT_STREAM
     }
   }
}
RoamPartner_EventSplitting
{
   ModuleName = FCT_EnhancedSplitting
   Module
   {
     Active          = True
     DataConnection  = ifw.DataPool.Login
     DefaultOutput   = SuspenseCreateOutput
     SystemBrands
     {
     XXX         = TAPOutput_XXX
      SUSP        = SuspenseCreateOutput
     }
   }
}
```

Sample Registry for Multiple Instances of Output Streams

```
Instances
{
TAPOutputStreaminstantiation
    {
    BlockName = Pipelines.TAPOutCollectPipeline.Output.OutputCollection.TAPOutput_XXX
    DataFile = ./RoamingPartnerConf.dat
    InstanceSpecificEntries
    {
      ModifyBlockName
      {
        Instance = [BlockName]
```

```
          UseColumn = TAPOUT_STREAM
         }
        ModifyOutputStreamSequencer
        {
            Instance = Module.Sequencer
            UseColumn = TAPOUT_SEQUENCER
        }
         ModifyRecipient
         {
             Instance = Module.Recipient
             UseColumn = VPLMN
          }
         ModifyCountryCode
         {
             Instance = Module.CountryCode
             UseColumn = COUNTRYCODE
          }
         ModifyDecimalPlaces
         {
             Instance = Module.DecimalPlaces
             UseColumn = DECIMALPLACES
          }
          ModifyOutputPath
          {
             Instance = Module.OutputStream.Module.OutputPath
             UseColumn = TAPOUT_PATH
          }
          ModifyOutputPrefix
          {
             Instance = Module.OutputStream.Module.OutputPrefix
             UseColumn = TAPOUT_PREFIX
          }
          ModifyTempPrefix
          {
             Instance = Module.OutputStream.Module.TempPrefix
             UseColumn = TMP_PREFIX
          }
          ModifyTempDataPath
          {
              Instance = Module.OutputStream.Module.TempDataPath
              UseColumn = TAPOUT_PATH
           }
           ModifyTempDataPrefix
           {
               Instance = Module.OutputStream.Module.TempDataPrefix
               UseColumn = TMP_DATA_PREFIX
            }
    }
}
TAPOutput_XXX
{
ModuleName  = OUT_GenericStream
ProcessType = TAPOUTCOLLECT_PIPELINE
EventType   = /event/delayed/session/telco/gsm
Module
{
   Grammar = ./formatDesc/Formats/TAP3-NG/TAP_0312_OutGrammar.dsc
   DeleteEmptyStream = False
   Sequencer = SEQ_GEN_TAPOUT_XXX
   Sender = PORTL
   Recipient = XXX
   CountryCode = XXX
```

```
   DecimalPlaces = XXX
   OutputStream
   {
     ModuleName = EXT_OutFileManager
     Module
     { OutputPath            = ./data/outcollect/tapout/XXX
       OutputPrefix          = CDPORTLXXX
       TempPrefix            = tmptest_XXX_
       TempDataPath          = ./data/outcollect/tapout/XXX
       TempDataPrefix        = test.XXX.tmp.
       TempDataSuffix        = .data
       UseInputStreamName    = [0,0]
       SequencerPrefix       = ""
       AppendSequenceNumber  = True
     }
   }
}

NRTRDEOutput_XXX
          {
            ModuleName  = XXX
            ProcessType = TAPOUTCOLLECT_PIPELINE
            EventType   = /event/delayed/session/telco/gsm

            Module
            {
              Grammar = ./formatDesc/Formats/TAP3/NRTRDE2_v01_OutGrammar.dsc
              DeleteEmptyStream = False
              Sequencer = SEQ_GEN_NRTRDEOUT_XXX

              Sender = EUR01
              Recipient = XXX

              OutputStream
              {
                ModuleName = EXT_OutFileManager
                Module
                {
                  OutputPath            = ./data/outcollect/nrtrdeout/XXX
                  OutputPrefix          = NREUR01XXX
                  TempPrefix            = tmptest_XXX_
                  TempDataPath          = ./data/outcollect/nrtrdeout/XXX
                  TempDataPrefix        = test.XXX.tmp.
                  TempDataSuffix        = .data
                  UseInputStreamName    = [0,0]
                  SequencerPrefix       = ""
                  AppendSequenceNumber  = True
                }
              }
            }
          }

NRTRDEOutputStreams
   {
   BlockName=Pipelines.TAPOutCollectPipeline.Output.OutputCollection.NRTRDEOutput_XXX
      DataFile = ./conf/RoamingPartnerConf.dat
      InstanceSpecificEntries
      {
        ModifyBlockName
        {
          Instance = [BlockName]
          UseColumn = NRTRDEOUT_STREAM
```

```
        }
        ModifyModuleName
        {
          Instance = ModuleName
          UseColumn = NRTDEOUTPUTSTREAMMODULE
        }
        ModifyOutputStreamSequencer
        {
          Instance = Module.Sequencer
          UseColumn = NRTRDEOUT_SEQUENCER
        }
        ModifyRecipient
        {
          Instance = Module.Recipient
          UseColumn = VPLMN
        }
        ModifyOutputPath
        {
         Instance = Module.OutputStream.Module.OutputPath
         UseColumn = NRTRDEOUT_PATH
        }
        ModifyOutputPrefix
        {
          Instance = Module.OutputStream.Module.OutputPrefix
          UseColumn = NRTRDEOUT_PREFIX
        }
        ModifyTempPrefix
        {
          Instance = Module.OutputStream.Module.TempPrefix
          UseColumn = TMP_PREFIX
        }
        ModifyTempDataPath
        {
          Instance = Module.OutputStream.Module.TempDataPath
          UseColumn = NRTRDEOUT_PATH
        }
        ModifyTempDataPrefix
       {
          Instance = Module.OutputStream.Module.TempDataPrefix
          UseColumn = TMP_DATA_PREFIX
        }
    }
}
```

LOG

Use the LOG module to manage and create your system log files:

Dependencies

The LOG module needs access to the log message table generated by the Controller in order to create the system log files. For information, see "Controller".

Registry Entries

Table 87-9 lists the LOG registry entries.

Table 87-9 LOG Registry Entries

Entry	Description	Mandatory?
FileName	Specifies the name of your system log file. If empty, the name will be built by the date.	No
FilePath	Specifies the path to your system log file.	No
FilePrefix	Specifies the log file prefix.	No
FileSuffix	Specifies the log file suffix.	No
LogLevel	Specifies the minimum severity limit. All messages greater or equal to the limit are logged. For example, enter **major** to log only major and critical error messages. Values are: • **critical** • **major** • **minor** • **warning** • **normal** • **debug** The default is **normal**, which means that all messages are logged. Using the **debug** setting returns additional debugging data.	No
MessageGroup	Specifies the message group name.	No
ProcessName	Specifies the process name.	No
ShowOriginator	Specifies whether to write the name of the module that emitted the message to the log file. **True** specifies to log the module name. This helps Oracle Technical support troubleshoot any problems. **False** specifies to not log the module name. The default is **False**.	No
SuppressErrors	Specifies any error messages to exclude from log files. For example, enter ERR_INSERTING_CLI to prevent those error messages from being logged.	No
WriteMessageKey	Specifies whether the module logs error codes. For example: ERR_FILE_NOT_FOUND. **True** specifies to write both the error code and error message to the log file. This helps technical support troubleshoot any problems. **False** specifies to write only the error message to the log file. The default is **False**.	No

Sample Registry Entry for the Process Log

```
ProcessLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            FilePath = /ifw/log/process
            FileName = process
            FilePrefix = log_
```

```
            FileSuffix = .log
            LogLevel = normal
            ProcessName = ifw
            SuppressErrors
            {
                INF_IGNORE_CLI
                ERR_INSERTING_CLI
            }
        }
    }
}
```

Sample Registry Entry for the Pipeline Log

```
PipelineLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            FilePath = /ifw/log/pipeline
            FileName = pipe2
            FileSuffix = .log
            LogLevel = minor
            SuppressErrors
            {
                INF_IGNORE_CLI
                ERR_INSERTING_CLI
            }
        }
    }
}
```

Sample Registry Entry for the Stream Log

```
OutputLog
{
    ModuleName = LOG
    Module
    {
        ITO
        {
            FilePath = /ifw/log/stream
            FilePrefix = stream_
            FileSuffix = .log
            LogLevel = normal
            SuppressErrors
            {
                ERR_SPEC_VERSION_INVALID
                ERR_RELEASE_VERSION_INVALID
            }
        }
    }
}
```

Semaphores

Table 87-10 lists the LOG Semaphores.

Table 87-10 LOG Semaphores

Entry	Description
FileName	Specifies the name of the log file. When you change the file name, the current log file is closed and renamed to file name plus timestamp. For example, the **process.log** file would be renamed **process_20030916130000.log**.
LogLevel	Specifies the minimum severity limit. The module logs all messages greater or equal to the limit. For example, enter **minor** to log only minor, major, and critical error messages. Values are: • **critical** • **major** • **minor** • **warning** • **normal** • **debug**
ShowOriginator	Specifies whether to write the name of the module that emitted the message to the log file. **True** specifies to log the module name. This helps Oracle Technical Support troubleshoot any problems. **False** specifies to not log the module name.
SuppressErrors	Specifies any error messages to exclude from the log file. For example, enter ERR_GETTING_DATADESCR to prevent those error messages from being logged.
WriteMessageKey	Specifies whether the module logs error codes. For example: ERR_FILE_NOT_FOUND. **True** specifies to write both the error code and error message to the log file. This helps Oracle Technical Support troubleshoot any problems. **False** specifies to write only the error message to the log file.

Sample Semaphores

```
ifw.ProcessLog.Module.ITO.FileName = process
ifw.ProcessLog.Module.ITO.LogLevel = minor
ifw.ProcessLog.Module.ITO.SupressErrors = ERR_GETTING_DATADESCR
```

Input Controller

Use the Input Controller to manage the input streams for its associated pipeline.

The Input Controller performs the following functions:

- Combines multiple CDR files into one transaction when configured to do so.
- Notifies the Transaction Manager (TAM) when a transaction begins.

You configure the Input Controller by editing the **Input** section of the registry file. For more information, see "Configuring the Input Section in the Registry".

Registry Entries

Table 87-11 lists the Input Controller registry entries.

Table 87-11 Input Controller Registry Entries

Entry	Description	Mandatory
InputModule	Subgroup for the Input module. See "INP_GenericStream".	Yes
UnitsPerTransaction	Specifies the number of CDR input files that make up a transaction. By default, each CDR file forms its own transaction. This parameter only affects processing within the pipeline, and the number of output files match the number of CDR input files. The default is **1**.	No

Sample Registry

```
Input
{
    UnitsPerTransaction = 2
    InputModule
    {
        ModuleName = INP_GenericStream
        Module
        {
            ...
        }
    }
}
```

NET_EM

The NET_EM module hosts an External Module (EM). This allows the NET_EM module to use the BRM API opcodes to transfer data between real-time rating opcodes and the real-time rerating, discounting, and zoning pipelines.

Registry Entries

Table 87-12 lists the NET_EM registry entries.

Table 87-12 NET_EM Registry Entries

Entry	Description	Mandatory
FieldName	Use this entry for real-time rerating. Specifies the field in the event flist to be used to route the event. By using the "**.**" notation, you can specify a field at any level in the flist. The field identified by FieldName must be of type POID or String.	No
FieldValue	Use this entry for real-time rerating. Specifies the value of the field identified by **FieldName**.	No
NumberOfRTPipelines	Number of real-time pipelines. **Note:** This number must match the **NumberOfInstances** entry.	Yes

Table 87-12 (Cont.) NET_EM Registry Entries

Entry	Description	Mandatory
OpcodeName	Specifies the opcode sending the event to NET_EM. For real-time discounting, use: PCM_OP_RATE_DISCOUNT_EVENT For real-time zoning, use: PCM_OP_RATE_GET_ZONEMAP_INFO For real-time rerating, use: PCM_OP_RATE_PIPELINE_EVENT	Yes
PipelineName	The pipeline to route the input to. Each real-time pipeline must have a unique name.	Yes
Port	Specifies the port number of the host machine running the NET_EM module.	Yes
Threads	Set this entry to the number of pipelines being managed by the NET_EM module. For example, if you have two pipelines, set this entry to 2.	Yes
UnixSockFile	Specifies the UNIX Sock file when the CM and the Pipeline Manager instance are running on the same machine.	Yes

Sample Registry Entry

```
DataPool
{
     RealtimePipeline
     {
          ModuleName = NET_EM
          Module
          {
               ThreadPool
               {
                    Port = 14579
                    Threads = 3
               }
               ReratingOpcode
               {
                    OpcodeName = PCM_OP_RATE_PIPELINE_EVENT
                    PipelineName = RealtimeReratingPipeline
                    NumberOfRTPipelines = 3
               }
          }
     }
}
```

Output Collection

Use the Output Collection module to handle output streams. See "About Configuring the Output Section in the Registry" and "Configuring EDR Output Processing".

Registry Entries

The only registry entries for the Output Collection configuration are the sections for each output stream, for example, OUT_DevNull, OUT_Reject, and OUT_GenericStream.

See the following:

- OUT_Reject
- OUT_DevNull
- OUT_GenericStream
- OUT_DB
- OUT_Realtime

Sample Registry

```
#-----------------------------------------------
# Output Section
#-----------------------------------------------
Output
{
   WriteDefaultEdr    = False
   DeleteEmptyFile    = True
     MaxErrorRates
     {
     }
     OutputCollection
     {
     #-----------------------------------------
     # The DevNull stream
     #-----------------------------------------
     DevNull
    ...{
```

Event Messages

Table 87-13 lists the Output Collection event messages.

Table 87-13 Output Collection Event Messages

Message	Description	Send/Receive
CMD_WRITE_LOG	An entry to the pipeline log has to be created.	Receive
REQ_STREAM_NUMBER	Determination of a specific stream number chosen by the first argument Name.	Receive from Output Collection. See "Output Collection".

Output Controller

Use the Output Controller to manage the output streams for its associated pipeline.

For more information, see "About Configuring the Output Section in the Registry".

Registry Entries

Table 87-14 lists the Output Controller registry entries.

Table 87-14 Output Controller Registry Entries

Entry	Description	Mandatory
MaxErrorRates	Subgroup for the maximum error rate entries. This section should list all error codes to monitor and their threshold values. For more information, see "Specifying the Maximum Errors Allowed in an Input File".	Yes
MultiThreading	Subgroup to configure multithreading in output processing.	No
MultiThreading.Active	Specifies whether to enable multithreading in output processing: • **True** enables multithreading. • **False** disables multithreading. This is the default.	Yes
MultiThreading.NumberOfThreads	Specifies the number of threads the Output Controller creates to manage the output streams for its associated pipeline. For optimum results, Oracle suggests that you set the number of threads to twice the average number of streams associated with an input EDR.	Yes
MultiThreading.BatchSize	Specifies the size of the batch in terms of number of EDRs: • **0** indicates that the Output Controller does not operate in a batch mode. • A value greater than **0** indicates that the Output Controller operates in the batch mode with the batch size equal to the specified value. Oracle suggests that **BatchSize** be greater than or equal to **BlockSize** if **BlockTransfer** is set to **True**; otherwise, **BatchSize** should be equal to the size of the output buffer. The **BatchSize** value should be directly proportional to **NumberOfThreads** and inversely proportional to the EDR enrichment rate.	Yes
OutputCollection	Subgroup for the Output Collection module entries. See: • About Configuring the Output Section in the Registry • Output Collection	Yes
OutputLog	Subgroup for the stream log entries.	Yes
SequenceGeneration	Specifies whether the pipeline generates one output file per CDR input file or one output file for an entire transaction. • **Unit** generates one output file per CDR input file. • **Transaction** generates one output file per transaction. For example, if you combine 5 CDR input files into one transaction, the pipeline creates only 1 output file. The default is **Unit**.	No
Sequencer	Specifies the name of the Sequencer for performing sequence checking. This Sequencer must be defined in the **SequencerPool** section of the registry file.	No

Table 87-14 (Cont.) Output Controller Registry Entries

Entry	Description	Mandatory
Statistic	Subgroup to control the statistics related to Pipeline Manager's EDR processing rate. You can view these statistics in the output logs and HTTP browser. See "About Configuring Statistics Information in the Output Section".	No

Sample Registry Entry for the Multithreaded Mode

```
Output
{
    MaxErrorRates
    {
        ERR_CUST_NOT_FOUND = 10
    }

   MultiThreading
   {
       Active = True
       NumberOfThreads = 5
       BatchSize = 500
    }
    Statistic
    {
        EdrCountCriteria = ALL
    }...
    OutputCollection
      ...
    OutputLog
    {
         ...
    }
    SequenceGeneration = Unit
    Sequencer = SequenceCheck1
    ...
}
```

Sample Registry Entry for the Single-Threaded Mode

```
Output
{
    MaxErrorRates
    {
        ERR_CUST_NOT_FOUND = 10
    }

    Statistic
    {
        EdrCountCriteria = ALL
    }...
    OutputCollection
      ...
    OutputLog
    {
         ...
    }
```

```
    SequenceGeneration = Unit
    Sequencer = SequenceCheck1
    ...
}
```

ParallelLoadManager

Use the ParallelLoadManager module to load your pipelines, data modules, and function modules in parallel.

Registry Entries

Table 87-15 lists the ParallelLoadManager registry entries.

Table 87-15 ParallelLoadManager Registry Entries

Entry	Description	Mandatory
Active	Specifies whether to load the pipelines, data modules, and function modules in parallel. • **TRUE** specifies to use parallel loading. • **FALSE** specifies to use sequential loading. If the entry is missing, parallel loading is disabled.	Yes
NumberOfThreads	Specifies the number of threads Pipeline Manager uses to load your pipelines, data modules, and function modules.	Yes

Sample Registry

```
ifw
{
   ...
   ParallelLoadManager
   {
      Active = TRUE
      NumberOfThreads = 4
   }
   …
}
```

Pipeline Controller

Use the Pipeline Controller to control its associated pipeline.

Registry Entries

Table 87-16 lists the Pipeline Controller registry entries.

Table 87-16 Pipeline Controller Registry Entries

Entry	Description	Mandatory
Active	Activates or deactivates processing in the pipeline. • **True** activates pipeline processing. • **False** deactivates pipeline processing.	Yes

Table 87-16 (Cont.) Pipeline Controller Registry Entries

Entry	Description	Mandatory
CountryCode	Specifies the valid country code for this pipeline. The default is **49** for Germany.	No
DataDescription	Speicifes the stream format description and mapping files See: • Configuring the Input DataDescription Registry Section • Configuring the Output DataDescription Registry Section	Yes
EdrFactory	Subgroup for the EDR Factory.	Yes
Functions	Subgroup for the function pool entries.	Yes
Input	Subgroup for the Input Controller. See: • Configuring the Input Section in the Registry • Input Controller.	Yes
Instances	Specifies multiple instances of a specific pipeline.	No
InternationalAccessCode	Specifies the international dial prefix. The default is **00** for Germany. **Note:** You can list multiple access codes by using a comma as a delimiter. For example: **00,001,002**.	No
InternationalAccessCodeSign	Specifies the international access code sign. The default is **+**.	No
InputBuffer	Subgroup that contains the entries for the buffer between the input module and function modules.	Yes
MobileCountryCode	Specifies the valid mobile country code for this pipeline. The default is **262** for Germany.	No
MultiThreaded	Specifies whether the pipeline uses multithreaded or single-threaded processing. The default is **True**. • **True** specifies multithreaded processing. • **False** specifies single-threaded processing.	No
NationalAccessCode	Specifies the dial prefix for national calls. The default is **0** for Germany.	No
NetworkDestinationCode	Specifies the network destination code, which identifies the home network for roaming calls. The default is **172** for D2.	No
NoOutputUsed	Specifies whether to load the Output module. Set this entry to **True** only when you are using single-threaded processing and the Input and Output modules are combined into one module. The default is **False**. • **True** specifies to *not* load the Output module. • **False** specifies to load the Output module.	No
Output	Subgroup that contains Output Controller entries. See: • About Configuring Output Processing • Output Controller.	Yes
OutputBuffer	Subgroup that contains the entries for the buffer between the function modules and the output module.	Yes
Pipeline_Name	Name of the pipeline.	Yes
PipelineLog	Subgroup that contains pipeline log entries.	Yes

Table 87-16 (Cont.) Pipeline Controller Registry Entries

Entry	Description	Mandatory
RejectStream	Specifies the name of the rejection stream. This stream must be defined in the output module. See: • Configuring Standard Recycling • Recycling EDRs in Pipeline-Only Systems	Yes
TransactionManager	Specifies the subsection for the Transaction Manager.	N/A

Sample Registry

```
Pipelines
{
    Pipeline01
    {
        Active = TRUE
        MultiThreaded = TRUE
        CountryCode = 49
        MobileCountryCode = 262
        NationalAccessCode = 0
        InternationalAccessCode = 00
        InternationalAccessCodeSign = +
        NetworkDestinationCode = 171
        RejectStream = XXX
        PipelineLog
        {
            ModuleName = LOG
            Module
            {
            }
        }
        InputBuffer
        {
            Size = 1000
        }
        OutputBuffer
        {
            Size = 1000
        }
        Input
        {
            InputModule
            {
                ModuleName = XXX
                ModuleStart = XXX
                Module
                {
                    ...
                }
            }
        } // Input
        Functions
        {
            FCI
            {
                FunctionPool
```

```
        {
            Function01
            {
                ...
            }
            Function02
            {
                ...
            }
        }
    }
} // Functions
Output
{
    ...
    Outputcollection
    {
        Output1
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        RejectOutput
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        Output2
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        DevNull
        {
            ModuleName = XXX
            ModuleStart = XXX
            Module
            {
                ...
            }
        }
        ...
    }
    OutputLog
    {
        ...
    }
    Sequencer
    {
        ...
```

```
                    }
                } #Output
            } #Pipeline01
        } #Pipelines
```

Sample Registry for Multiple Instances of a Pipeline

This sample shows how to configure multiple instances of a pipeline.

```
ifw
{
...
  Pipelines
  {
    Instances
    {
      RealtimeReratingPipelineGPRS
      {
        NumberOfInstances = 3
        InstanceSpecificRegistries
        {
          Entry1 = TransactionManager.BinaryLogFileName
          Entry2 = PipelineLog.Module.ITO.FileName
          Entry3 = OutputLog.FileName
          Entry4 = Functions.Standard.FunctionPool.EdrDump_Initial.Module.FileName
          Entry4 = Functions.Standard.FunctionPool.EdrDump_PostDiscounting.Module.FileName
          Entry5 = Functions.Standard.FunctionPool.EdrDump_PreDiscounting.Module.FileName
          Entry6 = Functions.Standard.FunctionPool.EdrDump_PostRating.Module.FileName
          Entry7 = Functions.Standard.FunctionPool.EdrDump_PreRating.Module.FileName
        }
...
```

Semaphore Entries

Table 87-17 lists the Pipeline Controller Semaphore entries.

Table 87-17 Pipeline Controller Semaphore Entries

Entry	Description
Active	Activates or deactivates processing in the pipeline.

Sample Semaphore Entry

```
ifw.Pipelines.ALL_RATE.Active = True
```

Event Messages

Table 87-18 lists the Pipeline Controller event messages.

Table 87-18 Pipeline Controller Event Messages

Message	Trigger	Parameter
EVT_PIPELINE_START	The pipeline was started.	Pipeline name from the registry.
EVT_PIPELINE_STOP	The pipeline was stopped.	Pipeline name from the registry.

Sequencer

Use the Sequencer to prevent Pipeline Manager from processing the same CDR file twice and to add tracking information to output streams.

Dependencies

When you configure the Sequencer to store state and log data in database tables, this module requires a connection to the Pipeline Manager database.

To assign a Sequencer to a pipeline, you must also configure the **Output** section of the registry file:

- To assign a sequence checker to a pipeline, use the **Sequencer** registry entry in the Output Controller module. For information, see "Output Controller".
- To assign a sequence generator to a pipeline, use the **Sequencer** registry entry in the output module. For information, see "OUT_GenericStream".

Registry Entries

Table 87-19 lists the Sequencer registry entries.

Table 87-19 Sequencer Registry Entries

Entry	Description	Mandatory
SequencerInstance	Specifies the name of the Sequencer instance.	Yes
Source	Specifies whether Sequencer state and log data are stored in files or in database tables. Values are: • **File** • **Database**	Yes
Controller	Subgroup that contains Controller entries	Yes
Controller.SequencerType	Specifies whether the Sequencer performs sequence checking or sequence generation: • **Check** configures the Sequencer to perform sequence checking. • **Generation** configures the Sequencer to perform sequence generation.	Yes
Controller.DatabaseConnection	Specifies a connection to the Pipeline Manager database.	Yes, only if **Source** = **Database**.
Controller.ReuseGap	Specifies whether the Sequencer assigns skipped sequence numbers to output files. • **True** directs the Sequencer to reuse skipped sequence numbers by assigning the skipped sequence numbers to other CDRs. • **False** directs the Sequencer to never reuse skipped sequence numbers. The default is **False**.	No
Controller.SequenceLength	Specifies the length of the incoming CDR file's sequence number. The default is **6**.	No

Table 87-19 (Cont.) Sequencer Registry Entries

Entry	Description	Mandatory
Controller.FileName	Specifies the name of the Sequencer state file. This file stores state information for one Sequencer instance. **Important:** You must create one state file for each Sequencer instance. Otherwise, the Sequencer fails.	Yes, only if **Source** = **File**
Controller.FilePath	Specifies the path to the Sequencer state file.	Yes, only if **Source** = **File**.
Controller.Log	Subgroup that contains Sequencer log file entries.	Yes
Controller.Log.FileName	Specifies the name of the Sequencer log file. **Important:** You must create one Sequencer log file for each Sequencer instance. Otherwise, the Sequencer fails.	Yes, only if **Source** = **File**
Controller.Log.FilePath	Specifies the path to the Sequencer log file.	Yes, only if **Source** = **File**
Controller.UseGapAtStartup	Specifies whether to add a gap for the skipped sequence numbers starting from 0. This entry is required only when the **SequencerType** field is **Check** and the **ReuseGap** field is **True**. You can use this entry even if you have set the **Seq Original Number** field to **0**. • **True**. This value directs the Sequencer to add a gap for the skipped sequence numbers starting from 0. • **False**. The default value. This value directs the Sequencer to never add a gap for the skipped sequence numbers.	No

Sample Registry for File Storage

```
SequencerPool
{
    SequencerInstance
    {
        Source = File

        Controller
        {
            SequencerType = Check
            ReuseGap = True
            SequenceLength = 7
            FileName = sequence.dat
            FilePath = /opt/portal/ifw/sequencer

            Log
            {
                FileName = sequence.log
                FilePath = /opt/portal/ifw/logs
            }
        }
    }
}
```

Sample Registry Entry for Database Storage

```
SequencerPool
{
```

```
    SequencerInstance
    {
        Source = Database

        Controller
        {
            SequencerType = Generation
            DatabaseConnection = DatabaseModule
            ReuseGap = False
            SequenceLength = 10
        }
    }
}
```

Database Tables

The Sequencer uses the following tables:

- IFW_PIPELINE
- IFW_SEQCHECK
- IFW_SEQLOG_IN
- IFW_SEQLOG_OUT

Transaction Manager

Use the Transaction Manager to coordinate the state of all transactional modules and components in one pipeline.

Dependencies

Requires a reference to the Transaction ID Controller. For information, see "Transaction ID Controller".

Registry Entries

Table 87-20 lists the Transaction Manager registry entries.

Table 87-20 Transaction Manager Registry Entries

Entry	Description	Mandatory?
BinaryLogFileName	Specifies the path and file name of the binary log file, which is used to persist and restore open transactions. **Important:** If you use multiple pipelines, you cannot use the same file for different pipelines.	Yes
RedoEnabled	Specifies whether the redo mechanism is enabled. **True** enables the redo mechanism. **False** disables the redo mechanism.	Yes

Table 87-20 (Cont.) Transaction Manager Registry Entries

Entry	Description	Mandatory?
SingleTransaction	Specifies whether only one pipeline transaction is allowed at a time. **True** specifies that only one pipeline transaction can be active at one time. The TAM blocks any new transactions from starting while a transaction is in progress. **False** specifies that multiple pipeline transactions can be active at one time.	Yes
WriteToLogEnabled	Specifies whether the Transaction Manager writes status information to the pipeline log file. **True** enables writing to the pipeline log file. **False** disables writing to the pipeline log file.	No

Sample Registry

```
Pipelines
{
    PipelineName
    {
        TransactionManager
        {
            RedoEnabled = True
            SingleTransaction = True
            BinaryLogFileName = ./
            WriteToLogEnabled = False
        }
    }
}
```

Transaction ID Database Generator

Use the Transaction ID Database Generator to store transaction IDs in database tables.

Dependencies

Requires a connection to the Pipeline Manager database.

Registry Entries

Table 87-21 lists the Transaction ID Database Generator registry entries.

Table 87-21 Transaction ID Database Generator Registry Entry

Entry	Description	Mandatory?
DataConnection	Specifies a connection to the Pipeline Manager database.	Yes

Sample Registry

```
TransactionIdController
{
```

```
    Source = Database
    Generator
    {
        DataConnection = ifw.DataPool.Login
    }
}
```

Database Tables

The TAM_TransIdDbGenerator module uses the IFW_TAM database table.

Transaction ID File Generator

Use the Transaction ID File Generator to store transaction IDs in a file.

Registry Entries

Table 87-22 lists the Transaction ID File Generator registry entries.

Table 87-22 Transaction ID File Generator Registry Entries

Entry	Description	Mandatory?
FileName	Specifies the path and file name of the Transaction ID Controller state file.	Yes
Increment	Specifies the number of transaction IDs that are cached.	Yes

Sample Registry

```
TransactionIdController
{
    Source = File
    Generator
    {
        FileName = /data/system/info/transIdInfo.dat
        Increment = 10
    }
}
```

Transaction ID Controller

Use the Transaction ID Controller to generate transaction IDs for all pipelines.

Registry Entries

Table 87-23 lists the Transaction ID Controller registry entries.

Table 87-23 Transaction ID Controller Registry Entries

Entry	Description	Mandatory?
Generator	Subgroup for the generator entries. See: • Transaction ID File Generator • Transaction ID Database Generator	Yes
Source	Specifies whether the Transaction ID Controller stores transaction IDs in files or database tables. Values are: • **File** • **Database**	Yes

Sample Registry for File Storage

```
TransactionIdController
{
    Source = File
    Generator
    {
        FileName = /data/system/info/transIdInfo.dat
        Increment = 10
    }
}
```

Sample Registry for Database Storage

```
TransactionIdController
{
    Source = Database
    Generator
    {
        DataConnection = ifw.DataPool.Login
    }
}
```

88
Pipeline Manager Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager utilities.

Database Loader

The **Database Loader** utility loads and unloads aggregation data into and from a database.

For information about aggregation, see "Setting Up Pipeline Aggregation".

Dependencies

This utility needs a connection to the DBC database module and the DBL library (**libDBLXXX.so**). See "Database Connect (DBC)".

Location

*pipeline_home***/tools**

where *pipeline_home* is the directory in which you installed Pipeline Manager.

Syntax

```
dbLoader -r registry [-f files] [-u]
```

Parameters

-r *registry*
Defines the registry file.

-f *files*
Defines the file pattern (regular expression).

-u
Undo mode.

Registry Entries

Table 88-1 lists the Database Loader registry entries.

Table 88-1 Database Loader Registry Entries

Entry	Description	Mandatory
BULKSIZE	Specifies the Oracle array size for bulk inserts (loadmode 2 and 3).	Yes
DIRECTIONMODE	Defines the selection order of the control files (1 file name, 2 sequence).	Yes
FILES.ARCHIVE.PATH	Specifies the path where the successfully loaded files are stored.	Yes
FILES.ARCHIVE.SUFFIX	Specifies the suffix of the successfully loaded data files.	Yes

Table 88-1 (Cont.) Database Loader Registry Entries

Entry	Description	Mandatory
FILES.BAD.PATH	Specifies the path where the bad files are stored.	Yes
FILES.BAD.SUFFIX	Specifies the suffix of the bad data files.	Yes
FILES.CONTROL.PATH	Specifies the path for the input aggregate control files.	Yes
FILES.CONTROL.SUFFIX	Specifies the suffix of the input aggregate control files.	Yes
FILES.DATA.PATH	Specifies the path for the input aggregate data files.	Yes
FILES.DATA.SUFFIX	Specifies the suffix of the input aggregate data files.	Yes
FILES.MERGE.PATH	Specifies the path where the source merge data files are stored.	Yes
FILES.MERGE.SUFFIX	Specifies the suffix of the source data files before merging/ sorting.	Yes
FILES.REJECT.PATH	Specifies the path where the rejected files are stored.	Yes
FILES.REJECT.SUFFIX	Specifies the suffix of the rejected data files.	Yes
LOADMODE	Specifies how to load data: • **1**: Single row updates and inserts. • **2**: Single row updates and bulk inserts. • **3**: Single row updates and bulk inserts. Before loading, the files can be merged or sorted and split into smaller pieces. Undo mode is always 1.	Yes
MAXSPLITLINES	Specifies the maximum number of lines per data file after splitting (loadmode 3).	Yes
ROLLBACKSEGMENT	Specifies which Oracle rollback segment to use when loading the database. How to set this entry depends on your database software setup. If your Oracle9i database uses automatic undo management, comment out or remove this registry entry. If your database does not use undo management, specify a rollback segment. The Oracle9i software provides an automatic undo management feature, which creates undo tablespaces rather than rollback segments for undo information. If you use this undo management feature *and* specify a rollback segment for the Pipeline Manager **Database Loader** utility, the utility fails when it attempts to load the database. To prevent this problem, don't specify a rollback segment.	No
SORTCMD	Specifies the external sort command (loadmode 3).	Yes
SORTING	Specifies a flag if files of identical structure should me merged and sorted (loadmode 3).	Yes
SORTMAXFILESIZE	Specifies the maximum destination size of the merged and sorted files (loadmode 3).	Yes
SORTTMPDIR	Specifies the path where sort stores temporary files (loadmode 3).	Yes
SPLITTING	Specifies whether to split data files before loading (reduce transaction size) (loadmode 3).	Yes

Sample Registry

```
DBLOADER
  {
    Active              = TRUE
    ProcessLoopTimeout  = 10
    QueueRequestTimeout = 0
    Instrumentation
    {
      #-------------------------------------------------------------
      # ProbeBroker registry entries.
      # ProbeInfoFilePath - The path that contains all probe
      # info files used by instrumented objects.
      #-------------------------------------------------------------
      ProbeBroker
      {
        ProbeInfoFilePath = ./instrumentation
      }
    }
    LogMessageTable
    {
      MessageFilePath   = ./etc
      MessageFileSuffix = .msg
    }
    DiagnosticDataHandler
    {
      DiagnosticFilePath = ./log
      DiagnosticFileName = diagnostic.dat
    }
    #
    # main parameter
    #
    DIRECTIONMODE   = 2
    LOADMODE        = 2
    BULKSIZE        = 100
    ROLLBACKSEGMENT = R04
    SORTING         = true
    SORTCMD         = sort
    SORTTMPDIR      = .
    SORTMAXFILESIZE = 2000000000
    SPLITTING       = true
    MAXSPLITLINES   = 40000
    #
    # database section
    #
    DataPool
    {
      Database
      {
        ModuleName = DBC
        Module
        {
          DatabaseName = $ORACLE_SID
          UserName     = AGGREGATOR
          PassWord     = password
          AccessLib    = oci11g72
          Connections  = 1
        }
      }
    }
    #
```

```
# File Section
#
FILES
{
  CONTROL
  {
    PATH    = ./data/aggregate/cntl
    SUFFIX  = .ctl
  }
  DATA
  {
    PATH    = ./data/aggregate/done
    SUFFIX  = .dat
  }
  REJECT
  {
    PATH      = ./data/aggregate/reject
    SUFFIX    = .rej
    THRESHOLD = 85
  }
  REJECT_HANDLE
  {
    PATH      = ./data/aggregate/reject
    SUFFIX    = .rej
    THRESHOLD = 85
  }
  ARCHIVE
  {
    PATH    = ./data/aggregate/archive
    SUFFIX  = .arc
  }
  BAD
  {
    PATH    = ./data/aggregate/bad
    SUFFIX  = .bad
  }
  MERGE
  {
    PATH    = ./data/aggregate/merge
    SUFFIX  = .mrg
  }
}
#
# log section
#
ProcessLog
{
  ModuleName = LOG
  Module
  {
    ITO
    {
      MessageFilePath = etc
      MessageFilePrefix = error
      MessageFileSuffix = error.msg
      FilePath = ./data/aggregate/log
      FileName = process
      FilePrefix = DBL_
      FileSuffix = .log
      ProcessName = dbLoader
      MessageGroup = DBLOADER
    }
```

```
        Buffer
        {
          Size = 1000
        }
      }
    }
  }
```

db2irules.pl

Use the **db2irules.pl** script to extract rule sets from the Pipeline Manager database into the Rule Set XML file.

> **Note:**
>
> This utility uses DBI and DBD drivers that are not part of the Pipeline Manager installation. You download these drivers from https://www.cpan.org and compile and install them separately.

Location

pipeline_home/**tools/IRules2Db/db2irules.pl**

> **Note:**
>
> Since there are dependencies between the **db2irules.pl** script and the **PerlParser.pm** XML library located in the same directory as the script, always run the script from this location.

Syntax

db2irules.pl [**-d**] [**-u**] *dbi:dcs file_path rule_set_id*

Parameters

If you start the **db2irules.pl** script without any parameters, a usage description and an example for each parameter are displayed.

-d
Deletes the specified rule sets from the database after you have extracted them. If you use this parameter, a transaction is opened with the database. If any of the rule set deletes fail, the entire delete sequence is rolled back to preserve database integrity. If all rule set tables are deleted successfully, the transaction is committed to the database.

-u
Creates a unique file name for the rule set, based on date and time. It uses the following format: *RULESET_yyyy-mm-dd_hh-mm-ss.xml*. Use this parameter to ensure that you do not override an existing XML file when extracting rule sets. If the file name for a rule set contains spaces, replace them with the underscore character (_).
Example:

```
db2irules.pl -u dbi:Oracle:orcl TAP3_VAL
```

dbi*:*dcs
Specifies the database connection string. This required parameter enables the script to access the database. The string is different for each database type. Example dcs for Oracle:

```
Oracle:orcl
```

> **Note:**
>
> The database connection string is the standard database access module for Perl scripts. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.

file_path
Specifies where you want to export the rule set. If you want to use the same directory in which the rule set is stored, use **./** as file path. If you don't set this parameter, the rule set is exported automatically to the current directory.

rule_set_id
Extracts only one specific rule set, which is identified by its unique ID. If you don't set this parameter, the **db2irules.pl** script will extract all rule sets from the database. If you use this parameter, you must use the *file_path* parameter. This *rule_set_id* refers to the IFW_RULESET.RULESET database field.

Diagnostic Data Handler

Use Diagnostic Data Handler to get data about Pipeline Manager after a crash, exception, critical error, or while it is running.

Registry Entries

Table 88-2 lists the Diagnostic Data Handler registry entries.

Table 88-2 Diagnostic Data Handler Registry Entries

Entry	Description	Mandatory
DiagnosticFilePath	Path to the log file that is created by Diagnostic Data Handler.	Yes
DiagnosticFileName	File name of the log file that is created by Diagnostic Data Handler.	Yes

Sample Registry

```
DiagnosticDataHandler
{
   DiagnosticFilePath = ./log
   DiagnosticFileName = diagnostic.dat
}
```

irules2db.pl

Use the **irules2db.pl** script to insert a rule set from the Validation Rules XML file into the Pipeline Manager database.

> **Note:**
>
> This utility uses DBI and DBD drivers which are not part of the Pipeline Manager installation. You download these drivers from https://www.cpan.org and compile and install them separately.

Location

*pipeline_home/***tools/IRules2Db/irules2db.pl**

> **Note:**
>
> Since there are dependencies between the **irules2db.pl** script and the **PerlParser.pm** XML library which is located in the same directory as the script. Always run the script from this location.

Syntax

```
irules2db.pl [-f] dbi:dcs rule_set_name backup_file_path
```

Parameters

If you start the **irules2db.pl** script without any parameters, a usage description and an example for each parameter are displayed.

dcs
The database connection string. This required parameter enables the script to access the database. The string is different for each database type. Example dcs for Oracle:

```
Oracle:orcl
```

> **Note:**
>
> The database connection string is the standard database access module for Perl scripts. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.

rule_set_name
Use this parameter to specify the name of the Rule Set XML file that you want to import to the database. This parameter supports fully qualified and relative path names.
Examples:

- **./tap3_val.xml**
- **/home/data/tap3_val.xml**
- **/../files/tap3_val.xml**
- **tap3_val.xml**

backup_file_path
Use this parameter to specify the path for storing the extracted rule set before it is deleted from the database and then after modification inserted from the Rule Set XML file into the database. Use this parameter with the **-f** parameter.

-f
This parameter forces the rule set into the database. The **irules2db.pl** script connects to the database and starts parsing the Rule Set XML file. When it finds the name of the rule set, it calls the export script that contains the **-u** and **-d** parameters. If the **db2irules.pl** script finished successfully, the **irules2db.pl** script continues parsing the XML file and imports the rule set to the database. If any of the rule set columns fail to be inserted, the **irules2db.pl** script rolls back the transaction and exits. If all columns are inserted into the database successfully, the rule set for the transaction is committed.

load_notification_event

Loads the notification XML file from Pipeline Manager into the BRM database. This allows BRM to notify customers when their balance has reached a threshold value during the batch rating process.

You must configure the Batch Controller to run this utility.

Location

BRM_home/**bin**

where, *BRM_home* is the directory in which you installed the BRM software.

Syntax

```
load_notification_event [-d] [-v] [-h] XML_file
```

Parameters

-d
Sets the log level to debug and outputs debug information into the log file for this process. If not set, only error-level information is output.

-v
Displays information about failed or successful processing as the utility runs.

-h
Displays syntax and parameters for this utility.

XML_file
The name and location of the XML file to load into the BRM database. This must be the last parameter listed on the command line.

Results

This utility notifies you when it successfully loads the XML file.

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

load_pin_rtp_trim_flist

Use this utility to specify account and service object fields to be included in the flist sent to a real-time rerating, discounting, or zoning pipeline. The main uses for this utility include:

- Improving system efficiency by removing (trimming) fields that Pipeline Manager doesn't use.
- Supporting custom iScripts and iRules in the real-time pipeline by adding fields to flists which are not included by default.

You can configure a different set of fields to be included in the flist based on event type.

Account object fields are included in the PIN_FLD_INHERITED_INFO substruct in the flist. Service object fields are included in the PIN_FLD_INHERITED_INFO.PIN_FLD_SERVICE_INFO substruct.

> **Note:**
>
> - You cannot load separate **/config/rtp/trim_flist** objects for each brand. All brands use the same object.
> - You can't remove fields from the PIN_FLD_INHERITED_INFO substruct or the subordinate PIN_FLD_INHERITED_INFO.PIN_FLD_SERVICE_INFO substruct.

You specify the list of required fields in an XML file (*field_list*.**xml**) and then load the file using the utility.

> **Note:**
>
> - If you use the utility to add new fields to the flist, you must update the input modules of the all pipelines to add the fields to the EDR container.
> - After you use the utility, you must restart BRM.

Location

BRM_home/**bin**

Syntax

```
load_pin_rtp_trim_flist -f field_list.xml [-v] [-d]
```

Parameters

-f *field_list*.xml
Specifies the XML file that describes which fields should be read. For a sample flist, see *BRM_home*/**sys/data/config/pin_config_rtp_trim_flist.xml**.

-v
Displays information about successful or failed processing as the utility runs.

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

LoadIfwConfig

Use this utility to extract data from or load data into the Pipeline Manager database. This enables you to:

- Migrate data from a legacy database to the Pipeline Manager database.
- Transfer data between Pipeline Manager databases; for example, from a test database to a production database. See "Transferring Data Between Pipeline Manager Databases".

> **Note:**
>
> The 7.4 version of the **LoadIfwConfig** utility is not backwards-compatible with previous versions of the utility. Any data exported by a previous version of the utility must also be loaded with that same version. In addition, any custom scripts or procedures that are dependent on the utility's functionality might need to be modified to work with the 7.4 version.

The **LoadIfwConfig** utility can run in these modes:

- *Non-interactive mode*: You use commands that batch several related parts of the extracting or loading process. You must enter a full command, including the utility name for each set of actions.
- *Interactive mode*: You issue a command for each step in the process of extracting or loading. After you enter interactive mode, the prompt changes to an angle bracket and commands are single words for performing particular actions. You can view a list of the change sets that will be extracted or loaded.

Location

pipeline_home/**bin**

Syntax: Non-Interactive Mode

```
LoadIfwConfig   {-rall [-t Modifidate] | -r [-t Modifidate] | -p [f] | -u | -I}
                [-c] [-nodep] -i InputFile [-o OutputFile] [-h] [-v]
```

Parameters: Non-Interactive Mode

-rall [-t *Modifidate*]
Extracts all objects from the Pipeline Manager database. This parameter does not require an input XML file.
Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified timestamp. Enter the time in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD* with the server time zone as the default.

-r [-t *Modifidate*]
Extracts from the database the objects listed in *InputFile*.
Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified time. Enter the time in the ISO-8601 format: *YYYY*-*MM*-*DD*T*hh*:*mm*:*ss* or *YYYY*-*MM*-*DD* with the server time zone as the default.

-p [f]
Deletes objects from the database.
Using the **f** parameter turns off the delete confirmation.

-u
Updates the Pipeline Manager database. Data is not actually updated in the database until it is committed with the **-c** parameter.

-l
Inserts data into the Pipeline Manager database. Data is not actually inserted into the database until it is committed with the **-c** parameter.

-c
Commits the data to the database. You use this command in conjunction with the **-u** and **-l** parameters.

-nodep
Suppresses any object dependency relationships that you configured in the *pipeline_home*/**tools/XmlLoader/CustomConfig.xml** file. This allows the utility to extract from the database only those objects that meet your criteria and to ignore any dependent objects. For more information about object dependencies, see "About Specifying to Extract Child and Dependent Objects".

-i *InputFile*
When extracting pipeline data by using the **-r** or **-rall** parameter, this is the name of the XML file that specifies the list of objects to extract from the source Pipeline Manager database.
When loading pipeline data by using the **-u** or **-l** parameter, this is the name of the XML file that contains the data you are loading into the destination Pipeline Manager database.
When deleting pipeline data by using the **-p** parameter, this is the name of the XML file that specifies the list of objects to delete from the Pipeline Manager database.

-o *OutputFile*
Specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named **default.out** in the current directory.

-h
Displays help about using the utility.

-v
Displays information about successful or failed processing as the utility runs.

Syntax: Interactive Mode

```
LoadIfwConfig [read InputFile] [write OutputFile] [retrieve_all [-t Modifidate]]
              [fetch [-t Modifidate]] [list] [delete] [commit] [update] [insert]
              [help] [nodep] [verbose on|off] [quit]
```

Parameters: Interactive Mode

read *InputFile*
Specifies to read the specified input file into internal memory.

write *OutputFile*
Specifies the output file to which the Pipeline Manager data is extracted. By default, the utility writes the output to a file named **default.out** in the current directory.

retrieve_all [-t *Modifidate*]
Extracts all objects from the Pipeline Manager database.

Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified time. Enter the time in the ISO-8601 format: *YYYY*-*MM*-*DD***T***hh*:*mm*:*ss* or *YYYY*-*MM*-*DD* with the server time zone as the default.

fetch [-t *Modifidate*]
Extracts from the database the objects listed in internal memory. You use this parameter after you use the **read** parameter.
Using **-t** *Modifidate* retrieves only pricing objects that were modified after the specified time. Enter the time in the ISO-8601 format: *YYYY*-*MM*-*DD***T***hh*:*mm*:*ss* or *YYYY*-*MM*-*DD* with the server time zone as the default.

list
Lists the current pipeline data stored in internal memory.

delete
Deletes from the database the objects listed in *InputFile*.

commit
Commits the data to the database. You use this command in conjunction with the **update** and **Insert** parameters.

update
Updates the Pipeline Manager database. Data is not actually updated in the database until it is committed with the **commit** parameter.

insert
Inserts data into the Pipeline Manager database. Data is not actually inserted into the database until it is committed with the **commit** parameter.

help
Displays help about using the utility.

nodep
Suppresses any object dependency relationships that you configured in the *pipeline_home***/tools/XmlLoader/CustomConfig.xml** file. This allows the utility to extract only those objects that meet your criteria and to ignore any dependent objects. For more information about object dependencies, see "About Specifying to Extract Child and Dependent Objects".

verbose [on | off]
Sets verbose information:

- **verbose on** displays the status of the command most recently run.

  Use the **ProcessLog** section of the registry file to specify the name and location of the file where debug messages are written.
- **verbose off** displays the status only if there is an error.

quit
Quits from the utility.

Results

If the **LoadIfwConfig** utility is successful, it displays a confirmation message. If unsuccessful, it displays errors.

Memory Monitor

Use the Memory Monitor module to warn you when available system memory is low and to shut down Pipeline Manager when memory reaches a specified threshold.

Registry Entries

Table 88-3 lists the Memory Monitor registry entries.

Table 88-3 Memory Monitor Registry Entries

Entry	Description	Mandatory
ScaleUnit	Specifies the unit for monitoring memory. • **P** specifies percentage. • **K** specifies Kilobytes. • **M** specifies MegaBytes.	Yes
ShutdownFreeMemLimit	Specifies the amount or percentage of remaining system memory that triggers Pipeline Manager to gracefully shut down. **Note:** For percentage, you must enter a value from 1 to 99 inclusive.	Yes
WarningFreeMemLimit	Specifies the amount or percentage of remaining system memory that triggers Pipeline Manager to issue a warning to the user. **Note:** For percentage, you must enter a value from 1 to 99 inclusive.	Yes

Sample Registry

```
ifw
{
    MemoryMonitor
    {
        ScaleUnit = P
        WarningFreeMemLimit = 10
        ShutdownFreeMemLimit = 5
    }
}
```

pin_container_to_stream_format

Use this utility to create EDR stream, input and output mapping, and input and output grammar files from an EDR container description file. FCT_CallAssembling then uses these files in the process of converting partially assembled call records to a new container description.

Location

BRM_home/**bin**

where *BRM_home* is the directory in which you installed BRM components.

Syntax

```
pin_container_to_stream_format -c container_description_filename -g
grammar_file_prefix -m mapping_file_prefix -s stream_file_prefix | -h
```

Parameters

-c *container_description_filename*
Specifies the container description file to use to generate a stream file and the mapping and grammar files. Replace *container_description_filename* with the container description file to use.

-g *grammar_file_prefix*
Creates the input and output grammar description files based on the container description file. Replace *grammar_file_prefix* with a prefix to add to the grammar filenames.

-m *mapping_file_prefix*
Creates the input and output mapping description files based on the container description file. Replace *mapping_file_prefix* with a prefix to add to the mapping filenames.

-s *stream_file_prefix*
Creates the stream description file based on the container description file. Replace *stream_file_prefix* with a prefix to add to the stream filename.

> **Note:**
>
> If you do not specify one or more of the **-g**, **-m**, or **-s** parameters, this utility generates the files using the container description filename as a prefix. However, if you specify these options, you must also specify their arguments. Otherwise this utility returns an error.

-h
Displays help for this utility.

Example

This example:

```
pin_container_to_stream_format -c containerDesc.dsc -g OLD_ -m OLD_ -s OLD_
```

Creates these files using the information in **containerDesc.dsc**:

- **OLD_Stream.dsc**
- **OLD_InGrammar.dsc**
- **OLD_OutGrammar.dsc**
- **OLD_InMap.dsc**
- **OLD_OutMap.dsc**

Results

The **pin_containter_to_stream_format** utility notifies you only if it encounters errors.

pin_recycle

Use this utility to search for failed EDRs in the BRM database and queue the EDRs for recycling or test recycling, or delete them. This utility can:

- Recycle calls from the same CDR file as part of the BRM standard recycling feature. For details, see "About the Standard Recycling Mechanism".
- Recycle all EDRs that contain the same recycle key as part of either Suspense Manager or standard recycling. For details, see "About Recycling Suspended EDRs after Rating Interruptions".
- Recycle all EDRs that have the same suspense reason code.

This utility calls the suspense manger opcodes to actually perform the recycling.

Location

BRM_home/**bin**

Syntax

```
pin_recycle [ -f CDR_file] [ -k recycle_key ] [ -d | -D| -r reason_code| -t ]
```

Parameters

-f *CDR_file*
Queues all the failed EDRs that arrived in a single CDR file. Pipeline Manager rates these calls as soon as it can.

-k *recycle_key*
Searches for and queues EDRs for rating that contain:

- The *recycle_key*, an application-specific string that is added to each EDR as it is suspended by Pipeline Manager. See "About Standard Recycling" for details.
- A status of **suspended**.

These EDRs are queued for rating by Pipeline Manager as soon as possible.

-d
Searches for and deletes all EDRs with a status of **succeeded** or **written off**.

-D
Searches for and deletes all EDRs with a status of **succeeded**, **written off**, or **suspended**.

-r *reason_code*
Searches for and recycles all EDRs that have the specified reason code.

-t
Specifies a test recycle. In test mode, **pin_recycle** creates a report about the processing, but does not make any changes to the database. Test results written to the directory and file you specified using the FCT_Suspense module **RecycleLog** registry entries. You must also set the FCT_Suspense **LogTestResults** registry entry for standard recycling implementations.

Results

This utility logs messages to **stdout**.

The following message is returned after you use **pin_recycle** to recycle EDRs:

```
pin_recycle tool, number_of_EDRs EDRs Submitted for Recycling
```

The following message is returned after you use **pin_recycle** to test recycle EDRs:

```
pin_recycle tool, number_of_EDRs EDRs submitted for test recycling
```

The following message is returned after you use **pin_recycle** to delete EDRs:

```
pin_recycle tool, number_of_EDRs suspended EDRs deleted
```

uninstaller

Use this utility to uninstall the BRM server software, client applications, and optional components from a single machine. If your BRM system is distributed among multiple machines, you must run the **uninstaller** utility on each machine.

This utility does not remove all BRM files and directories from your system or reverse changes made to your configuration files and database.

> **Note:**
>
> To use the **uninstaller** on Solaris, you must first install the latest patch for your version of Solaris.

Location

BRM_home/**uninstaller**

Syntax

```
uninstaller  -log BRM_home/uninstaller/uninst
             [ + | - | = ]product product_name
             [ + | - | = ]component component_name product_name
             read text_file_name
```

Parameters

-log *BRM_home*/uninstaller/uninst
Logs status and error messages to the **uninst** log file.

+
Registers the product or component to uninstall.

–
Points to the product or component to uninstall.

=
Verifies that the product or component is registered for uninstallation.

Commands

- **product** *product_name*

  Uninstalls the specified product. You can only uninstall one product at a time.

  The **Infranet.prod** file, located in the directory where you downloaded and extracted your BRM software, stores the names of all products installed on your system. *product_name* must match one of the names in this file.

  For example, to uninstall BRM, type:

  ```
  % uninstaller -log BRM_home/uninstaller/uninst -product Portal_Base
  ```

- **component** *component_name product_name*

  Uninstalls the specified component. You must specify the component name and the parent product.

  The **comps** directory, located in the directory where you downloaded and extracted your BRM software, lists the names of all components installed on your system. *component_name* must match one of the file names, minus the extension, in this directory.

  The **Infranet.prod** file, located in the directory where you downloaded and extracted your BRM software, stores the names of all products installed on your system. *product_name* must match one of the names in this file.

  For example, to remove the Connection Manager (CM) only, type:

  ```
  % uninstaller -log BRM_home/uninstaller/uninst -component CM Portal_Base
  ```

- **read** *text_file_name*

  Reads the text file and performs any batch operations specified in the text file.

Results

The **uninstaller** utility doesn't notify you whether it was successful or unsuccessful. You must look in your directory structure to see if your files were removed.

purge_np_data.p

Use this utility to purge existing records from the number portability data file that are older than a specified date and time. See "Purging and Reloading the Memory Records".

Location

*pipeline_home/***bin**

Syntax

```
purge_np_data.pl NP_FileName TimeStamp [-b backup_filename] [-n] [-help]
```

Parameters

NP_FileName
Specifies the name of the number portability data file that will be purged.

TimeStamp
Specifies the date prior to which all the number portability records are purged. After the data is purged, the number portability data file is updated with the purged data.
Format: *YYYYMMDDhhmmss*.

-b *backup_filename*
Specifies the name of the backup file that will contain the unpurged number portability records.

-n
Sorts in the ascending order of the CLI. Default sorting is in the ascending order of the time stamp.

-help
Displays the syntax and parameters for this utility.

Results

The **purge_np_data.pl** utility notifies you when it successfully purges the number portability data file. Otherwise, it displays an error message.

RoamingConfigGen64

Use this utility to retrieve the roaming partner data from the Pipeline Manager database and create the roaming configuration data file. The data file is used by the Instances module to configure multiple instances of sequencers, output streams, or system brands based on the template sections or entries in the roaming registry file.

Location

*pipeline_home/***bin**

Syntax

```
RoamingConfigGen64 -l database_access_library -s server_name [-d
database_name] -c operator_code [-o output_path] [-b base_path] [-h]
```

Parameters

-l database_access_library
The database access library.

-s *server_name*
Specifies the name of the host machine running the Pipeline Manager database.

-d *database_name*
Specifies the database name of the Pipeline Manager database. The default is an empty string (' ').

-c *operator_code*
Specifies the home network operator code. The default is **PORTL**.

-o *output_path*
Specifies the output path for the data file generated by the **RoamingConfigGen64** utility. By default, the data file is saved in the *pipeline_home***/conf/** directory.

-b *base_path*
Specifies the base path to the directory for Transferred Account Procedure (TAP) and Near Real Time Roaming Data Exchange (NRTRDE) output files. The default path is *pipeline_home***/data/outcollect/**.
For example, if the base path is *pipeline_home***/data/outcollect/**, the following new subdirectories are created in the *pipeline_home***/data/outcollect/** directory:

- **tapout/** for TAP output files
- **nrtrdeout/** for NRTRDE output files

-h
Displays the syntax and parameters for this utility.

> **Note:**
>
> When prompted, enter the database user name and password.

Example

```
RoamingConfigGen64 -l liboci10g6312d.so -s $ORACLE_SID -c EUR01
```

where:

- **liboci10g6312d.so** is the database access library.
- **$ORACLE_SID** is the database alias.
- **EUR01** is the home network operator code.

Results

The **RoamingConfigGen64** utility creates the roaming configuration data file. Otherwise, it displays an error message.

settlement_extract

Use this utility to retrieve roaming settlement information from the IC-Daily tables in the Pipeline Manager database. When Pipeline Manager rates roaming usage, it stores the amounts owed each roaming partner in the IC-Daily tables.

> **Note:**
>
> To ensure only unbilled events are extracted, before running this utility, you must close the bill run for each roaming partner account. You close the bill run by using Pricing Center or Pipeline Configuration Center (PCC).

This utility creates one file containing all settlement information stored in the Pipeline Manager database that has not already been extracted. The settlement information includes the amounts owed to each network that was used for roaming calls.

> **Note:**
>
> - This utility requires Perl version 5.004_00.
> - This utility uses DBI and DBD drivers which are not part of the Pipeline Manager installation. You download these drivers from https://www.cpan.org and compile and install them separately.
>
> For example:
>
> ```
> # setenv LD_PRELOAD /u01/app/oracle/product/817/JRE/lib/PA_RISC/
> native_threads/libjava.so
> ```

Location

BRM_home/**apps/uel**

Syntax

settlement_extract.pl [**-u**] **dbi:***dcs username password* [*filepath*]

Parameters

-u
Creates a unique file name for the new file using the current time. The format of the file name is:
"settlement_YYYY-MM-DD_hh-mm-ss.txt"

dcs
The database connection string. This required parameter enables the script to access the database. The string is different for each database type. Example dcs for Oracle:

```
Oracle:orcl
```

> **Note:**
>
> The database connection string is the standard database access module for Perl scripts. It defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used.

username
The database username.

password
The database password.

filepath
The location where the file should be written to. If you don't include this parameter, the file is written to the current directory.

Results

Creates a roaming settlement data file and reports success or displays an error.

stateconfigtool

Use this utility to load state configuration (**state.config**) files for use with the Pipeline Manager data migration feature.

> **Note:**
>
> Before you run **stateconfigtool**, make sure that the following files are listed in your system CLASSPATH environment variable:
>
> - **msbase.jar**
> - **msutil.jar**

Location

*pipeline_home***/tools/StateConfigTool**

Syntax

```
stateconfigtool -f file_name -d database_type -h host -n port -i database_id
```

Parameters

-f

The path and file name of the of the **state.config** file to be loaded. This file contains descriptions about changeset state transitions, such as currentState, nextState, and Action.

The default directory is *pipeline_home***/tools/StateConfigTool**.

-d

The database type. The supported database is **oracle**.

-h

The host name of the computer running the Pipeline Manager database.

-n

The port number used by the Pipeline Manager database.

-i

The database ID of the Pipeline Manager database.

Results

The utility loads the contents of the **state.config** file into the Pipeline Manager database. The states defined in the file become available in the Change Set Manager when it is restarted.

StopRapGen

The **StopRapGen** utility searches the database to collect information required by the Stop RAP Generator pipeline to create Stop Return Returned Account Procedure (RAP) files.

It retrieves information on the following:

- Transferred Account Procedure (TAP) files that were received by BRM and stored in the database more than seven days ago.
- Stop Return RAP files that were generated by BRM and sent more than seven days ago to the Visited Public Mobile Network (VPMN) operator.

> **Note:**
>
> The output from the **StopRapGen** utility is used by the Stop RAP Generator pipeline to generate the Stop Return RAP file.
>
> Use the **StopRapGen** utility along with the Stop RAP Generator pipeline.

Location

*pipeline_home***/bin**

where *pipeline_home* is the directory in which you installed Pipeline Manager.

Syntax

```
StopRapGen64 database_access_library server_name database_name path [prefix] [days]
```

Parameters

database_access_library
The database access library.

server_name
Specifies the name of the host machine running the Pipeline Manager database.

database_name
Specifies the database ID of the Pipeline Manager database.

path
Specifies the output directory of the flat file generated by the **StopRapGen** utility. This file is used by the Stop RAP Generator pipeline.

> **Tip:**
>
> The output directory for the **StopRapGen** utility should be the same as the input directory for the Stop RAP Generator pipeline.

prefix
Specifies the prefix to be added to the output flat file. The default prefix is **RC**.

days
Specifies the number of days to consider for generating a Stop Return RAP file. The default is 7, in accordance with the RAP standard.

Example

```
StopRapGen64 liboci10g6312d.so $ORACLE_SID '' ./data/stoprap/in
```

where:

- **liboci10g6312d.so** is the database access library.
- **$ORACLE_SID** is the database alias.
- ' ' is the empty string passed in as the database name.
- **.data/stoprap/in** is the output directory of the sample usage data for the **StopRapGen** utility (the flat file it generates). This is also the input directory of the Stop RAP Generator pipeline.

Results

The **StopRapGen** utility generates the input required by the Stop RAP Generator pipeline.

ZoneDBImport

The **ZoneDBImport** utility loads data in the IFW_STANDARD_ZONE table of the Pipeline Manager database.

This utility uses the following files:

- **Control File (zoneLoader.ctl**)

  The **zoneLoader.ctl** file controls how the data is loaded. It contains information about the table name, column datatypes, field delimiters, and so on.

  Initialize the infile variable with the path and file name of the file that contains the data to be imported.

- **Execution File (zoneLoader.pl)**

  Update the entries for the **DatabaseName** and **UserName** with the database name and user name of the current database.

Location

pipeline_home/**tools**

Syntax

```
./zoneLoader.pl
```

89

Pipeline Manager Configuration File Reference

This chapter lists the business_params settings used for configuring Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Pipeline Manager business_params Entries

Table 89-1 lists the Billing **business_params** entries.

Table 89-1 Billing business_params Entries

Name	Description	Implementation
AcctCycleDelayPeriod	Use when billing occurred after an event was rated by Pipeline Manager but before processing by RE Loader (that is, the event has not impacted the item). In such cases, RE Loader tries to locate the item from the next cycle if the event is created after billing in more than *X* days.	Not cached by the CM.
BatchRatingPipeline	Enables Account Synchronization to pass business events between **pin_rerate** and Pipeline Manager.	Cached by the CM. Restart the CM after changing this entry.
CacheResidenciesForBatchPipeline	Enables Pipeline Manager to selectively load accounts.	Cached by the CM and Pipeline Manager. Restart the CM and Pipeline Manager after changing this entry.

90
Pipeline Manager Opcode Reference

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) opcodes related to Pipeline Manager.

> **Caution:**
>
> - Always use the BRM API to manipulate data. Changing data in the database without using the API can corrupt the data.
> - Do not use SQL commands to change data in the database. Always use the API.

Account Synchronization FM Opcodes

The opcodes in Table 90-1 synchronize customer and service data with pipeline rating data.

Header File

Include the **ops/ifw_sync.h** header file in all applications that call these opcodes.

Opcode Index

Table 90-1 Account Synchronization FM Opcodes

Opcode	Description	Use
PCM_OP_IFW_SYNC_PUBLISH_EVENT	Passes events associated with this opcode in your system's event notification list to the policy opcode.	Limited
PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT	Policy for modifying the events passed to the standard opcode.	Recommended

PCM_OP_IFW_SYNC_PUBLISH_EVENT

Passes events associated with this opcode in your system's event notification list to the PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT policy opcode for processing.

By default, PCM_OP_IFW_SYNC_PUBLISH_EVENT passes events without modifying them.

PCM_OP_IFW_SYNC_POL_PUBLISH_EVENT

Modifies events included in account synchronization business events.

Events that trigger event notification for account synchronization make up the business events that the Account Synchronization Data Manager (DM) sends to Pipeline Manager. This opcode modifies specified triggering events before they are published to Pipeline Manager.

This opcode can also be used to filter out certain events that are not included in account synchronization (for example, an event that has only balance impacts for currency balance elements), thereby reducing the traffic in the listener queue.

If you pass an event that does not need any modification to this opcode, the opcode passes that event to the EAI framework for publishing.

Batch Suspense Manager FM Standard Opcodes

The opcodes listed in Table 90-2 manage batch files for suspended EDRs stored in the BRM database as **/suspended_batch** objects.

Header File

Include the **ops/batch_suspense.h** header file in all applications that call these opcodes.

Opcode Index

Table 90-2 Batch Suspense Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES	Deletes suspended batches from the BRM database. Available with Suspense Manager.	Recommended
PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES	Resubmits the batches which have been suspended by the pipeline.	Recommended
PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES	Writes off suspended batches.	Recommended

PCM_OP_BATCH_SUSPENSE_DELETE_BATCHES

Deletes suspended batches from the BRM database.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_BATCH_SUSPENSE_RESUBMIT_BATCHES

Initiates batch resubmission. During the resubmission process, suspended batches are sent back through their original rating pipelines. The Suspense Management Center calls this opcode when the user chooses to resubmit suspended batches.

PCM_OP_BATCH_SUSPENSE_WRITE_OFF_BATCHES

Writes off the batches which are at the “Suspended" stage because of some business rule. The GUI calls this opcode to write off the batches.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

Filter Set FM Standard Opcodes

This document describes the filter set opcodes listed in Table 90-3. These opcodes support BRM Pricing Center in providing separate products and discounts to the different market segments of your customer base. These opcodes allow you to divide your customers into market segments by filtering them based on criteria that you set in Pricing Center.

Header File

Include the **ops/filterset.h** header file in all applications that call these opcodes. See the discussion on header files in *BRM Developer's Guide*.

Opcode Index

Table 90-3 Filter Set FM Standard Opcodes

Opcode	Description	Use
PCM_OP_FILTER_SET_CREATE	Creates a new **/filter_set/product** object. See the discussion on creating filter sets in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_FILTER_SET_DELETE	Deletes a **/filter_set/product** object. See the discussion on deleting filter sets in *BRM Setting Up Pricing and Rating*.	Recommended
PCM_OP_FILTER_SET_UPDATE	Modifies a **/filter_set/product** object. See the discussion on updating filter sets in *BRM Setting Up Pricing and Rating*.	Recommended

PCM_OP_FILTER_SET_CREATE

Creates **/filter_set/product** objects, which store the list of system products and discounts that belong to a particular filter set. This opcode is called directly by Pricing Center.

See the discussion on creating filter sets in *BRM Setting Up Pricing and Rating*.

PCM_OP_FILTER_SET_DELETE

Deletes **/filter_set/product** objects. This opcode is called directly by Pricing Center.

See the discussion on creating filter sets in *BRM Setting Up Pricing and Rating*.

PCM_OP_FILTER_SET_UPDATE

Modifies the following data in **/filter_set/product** objects:

- The filter criteria
- The list of applicable system products and discounts

- The validity period

This opcode is called directly by Pricing Center.

See the discussion on updating filter sets in *BRM Setting Up Pricing and Rating*.

Suspense Manager FM Standard Opcodes

The opcodes listed in Table 90-4 manage suspended EDRs stored in the BRM database as **/suspended_usage** objects.

Header File

Include the **ops/suspense.h** header file in all applications that call these opcodes.

Opcode Index

Table 90-4 Suspense Manager FM Standard Opcodes

Opcode	Description	Use
PCM_OP_SUSPENSE_DEFERRED_DELETE	Deletes records for suspended EDRs after Revenue Assurance has been completed. Available with Suspense Manager.	Recommended
PCM_OP_SUSPENSE_DELETE_USAGE	Deletes records for suspended EDRs. Available with Suspense Manager.	Recommended
PCM_OP_SUSPENSE_EDIT_USAGE	Changes the contents of fields in suspended EDRs. Available with Suspense Manager.	Recommended
PCM_OP_SUSPENSE_RECYCLE_USAGE	Initiates EDR recycling.	Recommended
PCM_OP_SUSPENSE_SEARCH_DELETE	Deletes call records with a specific recycle key and a status of **succeeded** or **written off**.	Recommended
PCM_OP_SUSPENSE_SEARCH_EDIT	Changes fields in a large number of suspended records in one database operation.	Recommended
PCM_OP_SUSPENSE_SEARCH_RECYCLE	Recycles suspended EDRs. Available with Suspense Manager.	Recommended
PCM_OP_SUSPENSE_SEARCH_WRITE_OFF	Writes off a large number of suspended records in one database operation.	Recommended
PCM_OP_SUSPENSE_UNDO_EDIT_USAGE	Undoes edits to suspended EDRs. Available with Suspense Manager.	Recommended
PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE	Writes off suspended EDRs.	Recommended

PCM_OP_SUSPENSE_DEFERRED_DELETE

Deletes EDRs in a written off state or succeeded state. This opcode is scheduled to run at a later time to ensure Revenue Assurance.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_DELETE_USAGE

Deletes EDRs in a written off state or succeeded state.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_EDIT_USAGE

Changes the contents of EDR fields for a suspended call record. The Suspense Management Center calls this opcode to edit a suspended call record.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_RECYCLE_USAGE

Initiates EDR recycling. During recycling, suspended EDRs are sent back through their original rating pipelines. The Suspense Management Center calls this opcode when the user chooses to recycle suspended EDRs.

PCM_OP_SUSPENSE_SEARCH_DELETE

Deletes call records with a status of **succeeded** or **written off** that match criteria specified in the input flist. You can specify the following criteria:

- A recycle key
- A CDR file
- A search template

This opcode can also delete a **suspended** call record if PIN_FLD_MODE is set correctly.

PCM_OP_SUSPENSE_SEARCH_EDIT

This opcode makes changes to a large number of suspended records that meet the criteria specified in the input template.

PCM_OP_SUSPENSE_SEARCH_RECYCLE

Searches for and queues suspended call records for recycling based on criteria specified in the input flist. You can specify the following criteria:

- A recycle key
- A CDR file

- A search template

PCM_OP_SUSPENSE_SEARCH_WRITE_OFF

This opcode writes off a large number of suspended records that match the search criteria in the input flist.

PCM_OP_SUSPENSE_UNDO_EDIT_USAGE

Undoes edits to suspended call records used by Suspense Manager. This opcode is called by Suspense Management Center to perform the undo edit action. It replaces the value of a field in a suspended call record with the value in that field before the last edit was made.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

PCM_OP_SUSPENSE_WRITTEN_OFF_USAGE

Writes off suspended EDRs. When a suspended EDR is written off, they cannot be edited or recycled.

> **Note:**
>
> This opcode is available to Suspense Manager customers only.

91
Event Notification Definitions

This chapter provides a brief description of each Oracle Communications Billing and Revenue Management (BRM) notification event and includes links to the notification event specifications. See "Using Event Notification" in *BRM Developer's Guide* for more information.

Event Notification Definitions

Table 91-1 lists the BRM event notification definitions and descriptions.

Table 91-1 Event Notification Definitions

Event Notification	Description
/event/notification/activity/out_of_order	Generated when an out-of-order event is detected. When the notification event is detected, Pipeline Manager automatically rerates events.
/event/notification/amt/HoldCDRProcessing	Generated when Account Migration Manager begins migrating a group of accounts from one database schema to another. This event notifies the account-router Pipeline Manager that it needs to suspend all EDRs for the specified accounts.
/event/notification/amt/MigrateAcct	Generated after Account Migration Manager successfully migrates a group of accounts from one database schema to another. This event notifies the account-router Pipeline Manager that it needs to update the POIDs for the specified list of accounts.
/event/notification/amt/MigrateDestination	Generated after Account Migration Manager successfully migrates a group of accounts from one database schema to another. This event notifies the destination Pipeline Manager that it needs to update the account information stored in cache.
/event/notification/amt/MigrateSource	Generated after Account Migration Manager successfully migrates a group of accounts from one database schema to another.
/event/notification/amt/ResumeCDRProcessing	Generated after both Account Migration Manager successfully migrates a group of accounts and all Pipeline Manager instances successfully update their account information. This event notifies the account-router Pipeline Manager that it can begin processing all suspended and new EDRs for the specified list of accounts.
/event/notification/suspense/batch_delete	Generated when a suspended batch is purged.
/event/notification/suspense/batch_resubmit	Generated when a suspended batch is submitted for recycling.
/event/notification/suspense/batch_writeoff	Generated when a suspended batch is written off.
/event/notification/suspense/delete	Generated when a suspense record is deleted.
/event/notification/suspense/edit	Generated when a suspense record is modified.
/event/notification/suspense/recycle	Generated when a suspense record is recycled.
/event/notification/suspense/writeoff	Generated when a suspense record is written off. F

92

Revenue Assurance Manager Reports

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) reports that support batch rating-related Revenue Assurance Manager features.

Revenue Assurance Rating Summary Report

> **Note:**
>
> You must upload the **RevAssuranceRating.source** file that is in the **StoredProcedures** folder to the database after the report is installed.

The Revenue Assurance Rating Summary report (**RevAssuranceRatingSummary.rtf**) shows revenue assurance data collected from pipeline rating. This report returns information for a specified time period.

There are different subreports created for different rating **/process_audit** objects.

The SQL query for this report is in the *BIP_home***xmlp\XMLP\Reports\BRM Reports\Revenue Assurance Reports\StoredProcedures\RevAssuranceRating.source** file, where *BIP_home* is the directory in which Oracle Business Intelligence (BI) Publisher is installed.

No charts are available for this report.

Revenue Assurance Rating Summary Report Parameters

You can change the following parameters to modify the output of the Revenue Assurance Rating Summary report:

- Start Date (process start date or date-time)
- End Date (process end date or date-time)
- Trans Start Date (transaction start date or date-time)
- Trans End Date (transaction end date or date-time)
- Flow
- Control Point
- Service Type

> **Note:**
>
> If you want to generate a report for the service types that are not in the list of the default service types, add the service type in the list of values in BI Publisher. For more information on adding the list of values, see the BI Publisher documentation.

- Batch Type

Table 92-1 Revenue Assurance Rating Summary Parameters

Parameter	Description	Valid Values
Flow	A collection of linked control points in a pipeline.	Any valid flow name. Default: not specified
Control point	An instance of FCT_AggreGate that you configure in your pipeline to collect revenue assurance data. For example, CP_Afterrating.	Any valid control point names or **ALL**. Default: **ALL**
Service Type	A commodity sold by your company and that your customers can purchase and use. For example, SMS or TEL.	Any valid service type names. Default: not specified
Batch Type	The type of batch that the EDRs belong to. • **0** for normal • **1** for rerating • **2** for recycling • **3** for write-off	The batch type numbers. Default: not specified

Revenue Assurance Rating Detail Report

> **Note:**
>
> You must upload the **RevAssuranceRating.source** file that is in the **StoredProcedures** folder to the database after the report is installed.

The Revenue Assurance Rating Detail report (**RevAssuranceRatingDetail.rtf**) shows detailed statistics for the revenue assurance data collected from pipeline rating. By default, the statistics are organized by EDR status. For each status, this report displays the following:

- Volume sent
- Volume received

The SQL query for this report is in the *BIP_home***xmlp\XMLP\Reports\BRM Reports\Revenue Assurance Reports\StoredProcedures\RevAssuranceRating.source** file, where *BIP_home* is the directory in which BI Publisher is installed.

No charts are available for this report.

Revenue Assurance Rating Detail Parameters

You can change the following parameters to modify the output of the Revenue Assurance Rating Detail report:

- Start Date (process start date or date-time)
- End Date (process end date or date-time)
- Trans Start Date (transaction start date or date-time)
- Trans End Date (transaction end date or date-time)
- Flow
- Control Point
- Service Type
- Batch Type
- Batch ID
- Successful records
- Failed records
- Written Off records
- Dynamic Data
- Revenue Assurance Scope

Table 92-2 Revenue Assurance Rating Detail Parameters

Parameter	Description	Valid Values
Flow	A collection of linked control points in a pipeline. For example, Flow1. **Important**: This field is mandatory.	Any valid flow name. Default: not specified
Control point	An instance of FCT_AggreGate that you configure in your pipeline to collect revenue assurance data. For example, CP_Afterrating. **Important**: This field is mandatory.	Any valid control point names or **ALL**. Default: **ALL**
Service Type	A commodity sold by your company and that your customers can purchase and use. For example, SMS or TEL.	Any valid service type names. Default: not specified
Batch Type	The type of batch that the EDRs belong to. • **0** for normal • **1** for rerating • **2** for recycling • **3** for write-off	The batch type numbers. Default: not specified
Batch ID	Batch ID of the EDRs. For example, MED1 or MED2. **Important**: This field is mandatory.	Any valid batch ID. Default: not specified
Successful Records	The status of successful EDRs.	The name given to define the successful EDRs in pipeline rating. Default: **Successful**

Table 92-2 (Cont.) Revenue Assurance Rating Detail Parameters

Parameter	Description	Valid Values
Failed Records	The status of failed EDRs.	The name given to define the failed EDRs in pipeline rating. Default: **Suspense**
Written Off Records	The status of written-off EDRs.	The name given to define the written-off EDRs in pipeline rating. Default: **Written-off**
Dynamic Data	Determines whether the revenue assurance data is dynamic or static. Select **YES** to show dynamic data. Select **NO** to display static data. The static data displays data on the call data records (CDRs) rated from the mediation batches. The dynamic data displays data on the CDRs that are rejected, recycled, or rerated.	**YES** or **NO**. Default: **NO**
Revenue Assurance Scope	Refers to the Batch Type parameter.	Batch rating or rerating. Default: not specified