Oracle® Communications Billing and Revenue Management

Pipeline Manager Administration Guide

Release 15.1
F93240-01
April 2025

ORACLE

Oracle Communications Billing and Revenue Management Pipeline Manager Administration Guide, Release 15.1

F93240-01

Contents

6 Migrating Accounts with the Pipeline Manager Running

7 Pipeline Manager Error Messages

Preface

This book describes system administration tasks for Oracle Communications Billing and Revenue Management Pipeline Manager.

Audience

This guide is intended for system administrators who maintain and manage the system.

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 customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

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.

1

Pipeline Manager System Architecture

Learn about the Oracle Billing and Revenue Management Pipeline Manager system architecture.

Topics in this document:

About the Pipeline Manager System Architecture

Pipeline Manager is used for rating and discounting events in batch and real-time.

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 1-1 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 1-1 Billable Event Rating by Pipeline Manager and Storage in BRM Database

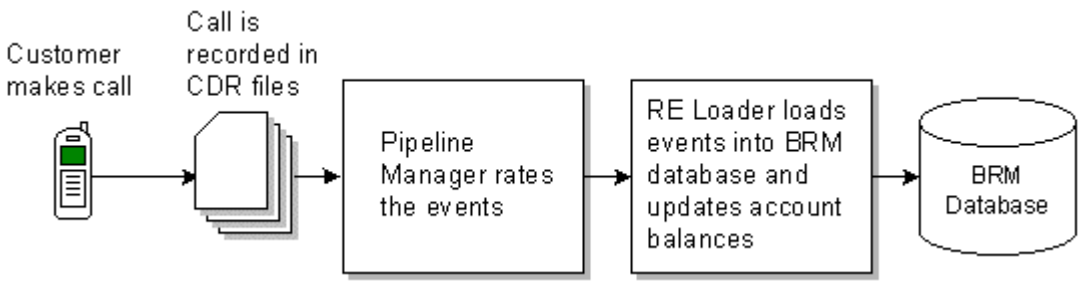
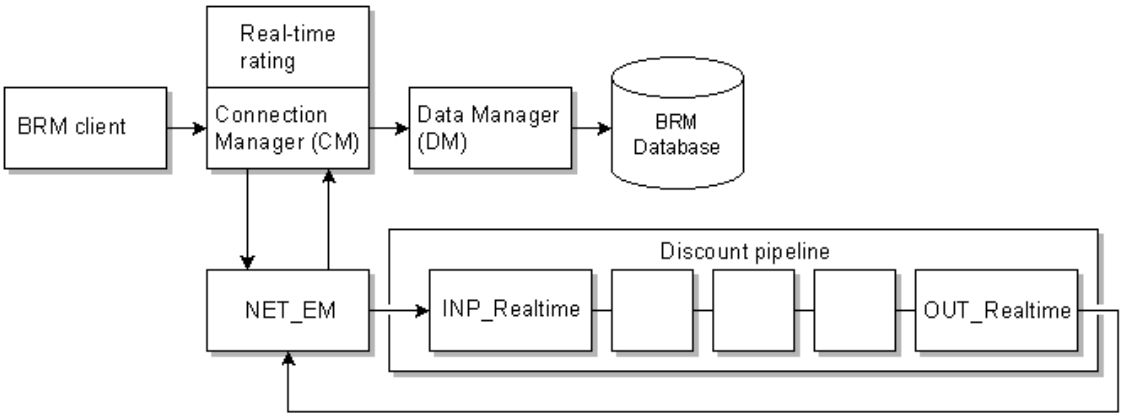


Figure 1-2 shows how real-time discounting works.

Figure 1-2 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 pricing 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.

2

Configuring Pipeline Manager

Learn how to manage Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager framework components and pipelines.

Topics in this document:

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 2-1.

Table 2-1 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_name
                        Module
                        {
                             Entry = value
...
```

The entries listed in Table 2-2 are a combination of required text and text that you define.

Table 2-2 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**.
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_name*	**ModuleName** is the entry. You must use **ModuleName**. *Module_name* 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 library 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_name
        Module
        {
            Entry = value
```

The entries listed in Table 2-3 are a combination of required text and text that you define.

Table 2-3 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**.

Table 2-3 (Cont.) Pipeline Registry DataPool Section Entries

Entry	Description
ModuleName = *Module_name*	**ModuleName** is the entry. You must use **ModuleName**. *Module_name* 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 library 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 2-4 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 2-4 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 Rogue Wave buffer. See "Using Rogue Wave Buffers".

> **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.

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 2-5, goes in the **Buffer**, **InputBuffer**, or **OutputBuffer** registry section.

Table 2-5 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 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 2-6 to specify where and how often the Controller checks for semaphore files.

Table 2-6 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 processed 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 terminated.

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 Log Files and Log 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 2-7, which are automatically created by the Pipeline Manager installer.

Table 2-7 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".
> - 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 2-8.

Table 2-8 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 2-9 control Pipeline Manager log files.

Table 2-9 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.
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.

Table 2-9 (Cont.) Pipeline Manager Log File Registry Entries

Entry	Module	Log file	Description
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.
```

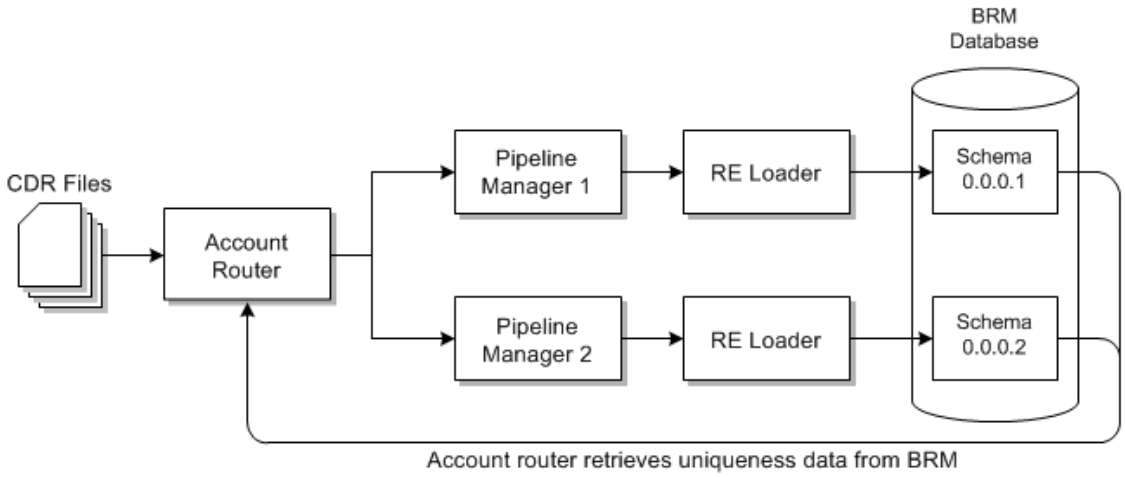
Using Pipeline Manager with Multiple Database Schemas

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 Oracle 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 2-1 shows how FCT_AccountRouter manages incoming CDRs.

Figure 2-1 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 2-10.

Table 2-10 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 2-11 describes the files and scripts used for controlling pipelines.

Table 2-11 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.

Table 2-11 (Cont.) Pipeline Manager Administration Perl Scripts

Semaphore and Perl script file names	Description
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.

3
Starting and Stopping Pipeline Manager

Learn how to start and stop Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager framework components and pipelines.

Topics in this document:

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 3-1 describes the files and scripts used for starting and stopping pipelines.

Table 3-1 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.

Table 3-1 (Cont.) Scripts for Starting and Stopping Pipelines

Semaphore and Perl script file names	Description
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
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).

4

Monitoring Pipeline Manager

Learn how to monitor Oracle Communications Billing and Revenue Management Pipeline Manager.

Topics in this document:

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 previously compiled data files.

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.

> **Note:**
>
> 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.

5

Optimizing Pipeline Manager Performance

You can use multiple tools and techniques to optimize Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager performance.

Topics in this document:

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:

- (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.)
> - 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**.

Configuring Threads for DAT_BalanceBatch Connections

Use the following DAT_BalanceBatch registry entries 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 5-1.

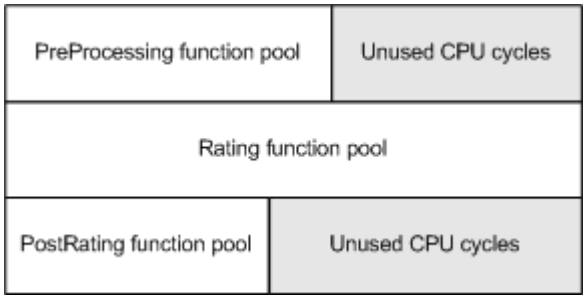
Table 5-1 Example 1 Module Latencies Summary

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 5-1 shows that about a third of the CPU cycles used by the function pool threads are idle.

Figure 5-1 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 {...}
```

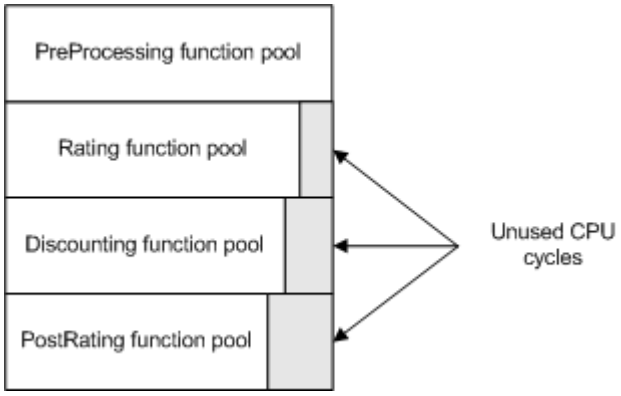
The latency table now appears as shown in Table 5-2.

Table 5-2 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 5-2.

Figure 5-2 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 5-3.

Table 5-3 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
        }
```

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.
> - 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:

   ```
   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.
> - 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.

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 for 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 window 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 ensures 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.

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.

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 than 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 binary.
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.

6

Migrating Accounts with the Pipeline Manager Running

Learn how Oracle Communications Billing and Revenue Management (BRM) migrates accounts when Pipeline Manager is running. Also, learn how to configure your BRM system to migrate accounts when Pipeline Manager is running.

Topics in this document:

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

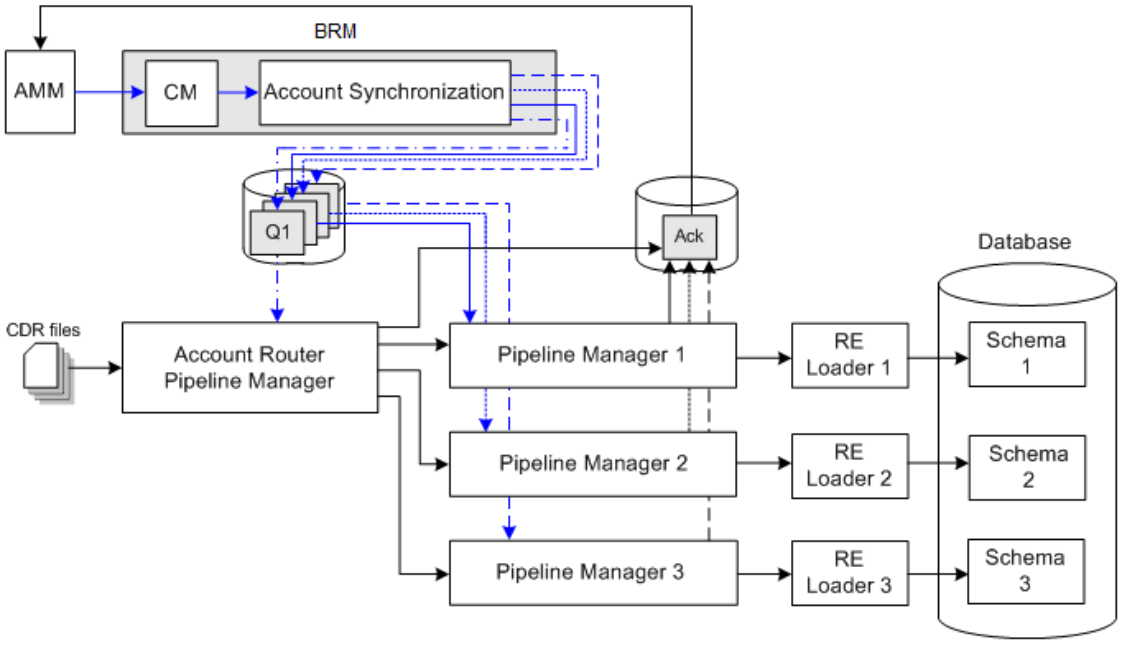
By default, AMM does not support migration when your pipelines are running. You specify whether AMM can migrate accounts while the Pipeline Manager is online by using the **controller**_*N*_**event_generation** parameter in the AMM **Infranet.properties** file.

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 6-1 shows AMM interaction with the Pipeline Manager.

Figure 6-1 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 AMM Business Events

AMM generates the five business events listed in Table 6-1 to notify the account router, source, and destination instances of the Pipeline Manager when account migration occurs.

Table 6-1 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

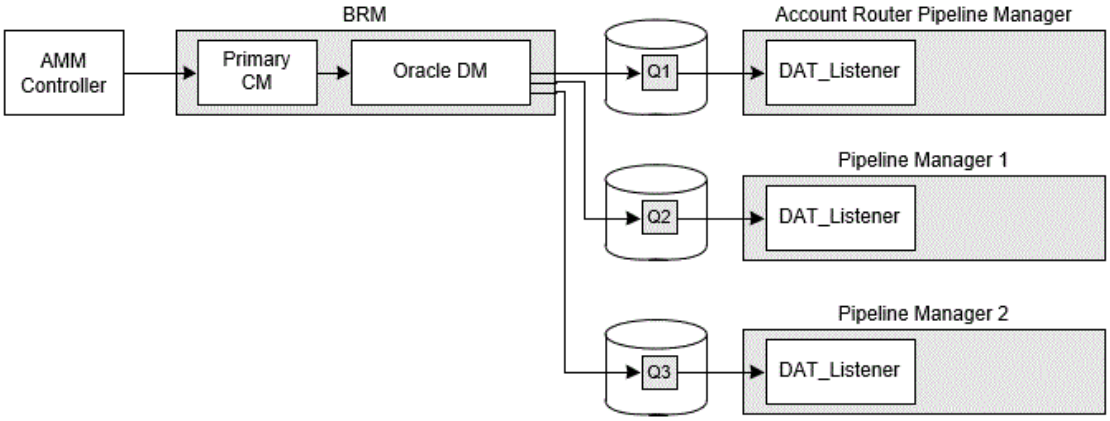
AMM sends business events to the pipelines by using a series of AQ database queues. Each instance of the Pipeline Manager contains its own queue, which is dedicated to receiving business events from BRM and AMM.

> **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 Oracle DM.
3. The Oracle DM uses its **aq_event_map** 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 6-2 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 AQ database queue for each instance of the Pipeline Manager.
- Configuring Oracle DM to publish AMM business events to your queues.
- Configuring each instance of the Pipeline Manager to dequeue AMM business events from its associated queue.

Figure 6-2 shows the AMM business events process described. See "Configuring AMM to Send Business Events to Your Pipelines".

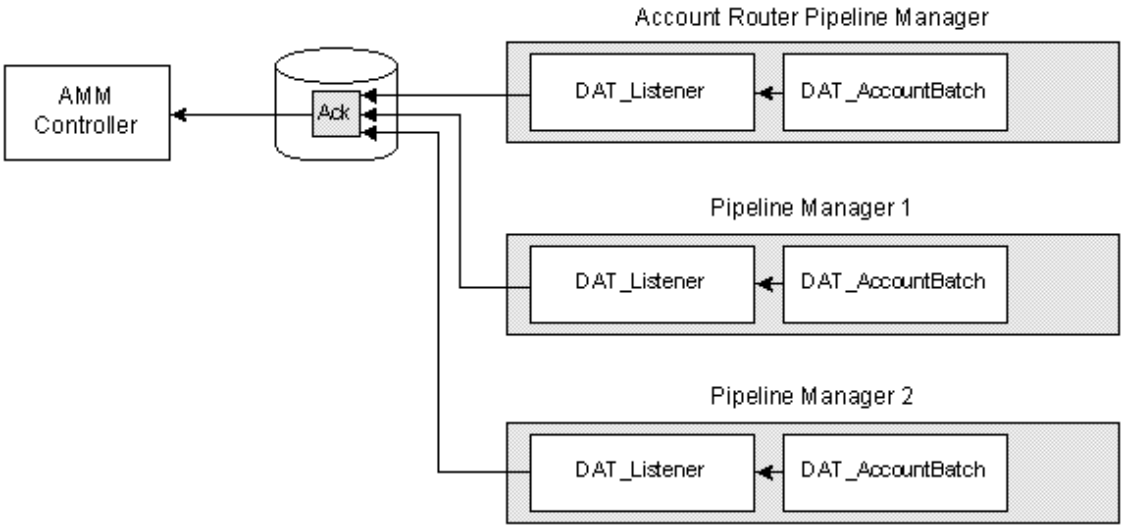
About Sending Acknowledgments to AMM

Each instance of the Pipeline Manager sends acknowledgment events to AMM by using a dedicated acknowledgment queue as shown in Figure 6-3.

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 6-3 Sending Pipeline Acknowledgement 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 Notifying the Pipelines About Account Migration

AMM notifies your pipelines about account migration by sending a series of business events through the Oracle DM and AQ database queues.

About Acknowledgment Events

Each instance of the Pipeline Manager generates acknowledgment events when the following actions listed in Table 6-2 occur.

Table 6-2 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.
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 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 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:

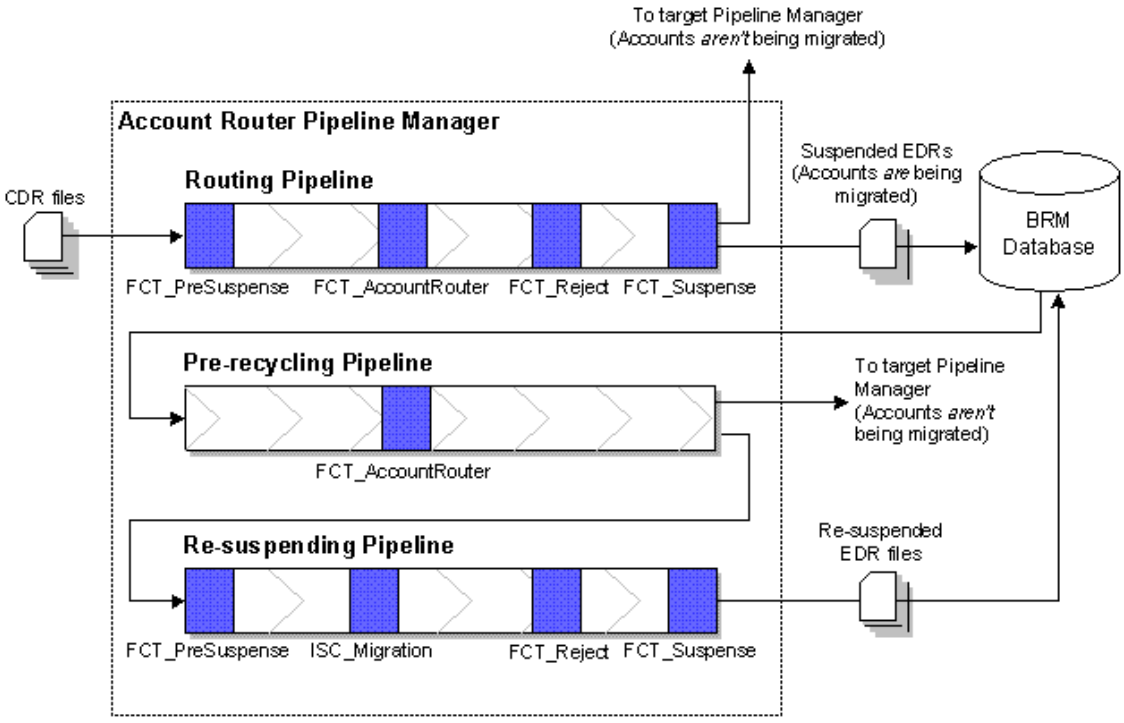
- Suspends EDRs targeted for accounts that are undergoing migration. See "About Suspending Call Records".
- Recycles previously suspended EDRs. See "About Reprocessing Suspended Call Records".

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 6-4 shows the necessary pipelines and account router data pool.

Figure 6-4 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 Acknowledgment Events

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 route to 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 AQ database 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 Synchronization of AMM Business Events
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 Oracle DM, 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 Synchronization of AMM Business Events

You configure BRM to send AMM business events to the appropriate instance of the Pipeline Manager by performing the following:

1. Configuring BRM to synchronize account data with Pipeline Manager. See "Synchronizing Account Data between BRM and Pipeline Manager" in *BRM System Administrator's Guide*.
2. Creating Oracle Advanced Queuing (AQ) database 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 AQ Database Queues for AMM Business Events".
3. Mapping the AMM business events to your AQ database queues. See "Mapping AMM Business Events to Your Queues".

Configuring Your AQ Database Queues for AMM Business Events

The account synchronization process uses a set of AQ database queues to send both BRM events and AMM business events to your pipelines.

Each instance of Pipeline Manager must have its own queue. For example, if your system contains three BRM database schemas, you must have 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 create a separate queue for any external applications requiring account migration notifications.

If your system does not already contain a queue for each instance of the Pipeline Manager, you can create additional queues by using the **pin_publish_aq_oracle.pl** utility.

For information about creating and configuring your AQ database queues, see "Configuring Your AQ Database Queues" in *BRM System Administrator's Guide*.

Mapping AMM Business Events to Your Queues

You map which types of events the Oracle DM sends to your queues by using the **aq_event_map** file (*BRM_home*/**sys/dm_oracle/aq_event_map**). This file lists all queues in your system and the events to route to each one.

You configure the **aq_event_map** file to map AMM business events, in addition to your BRM events, to each of your queues by using the following syntax:

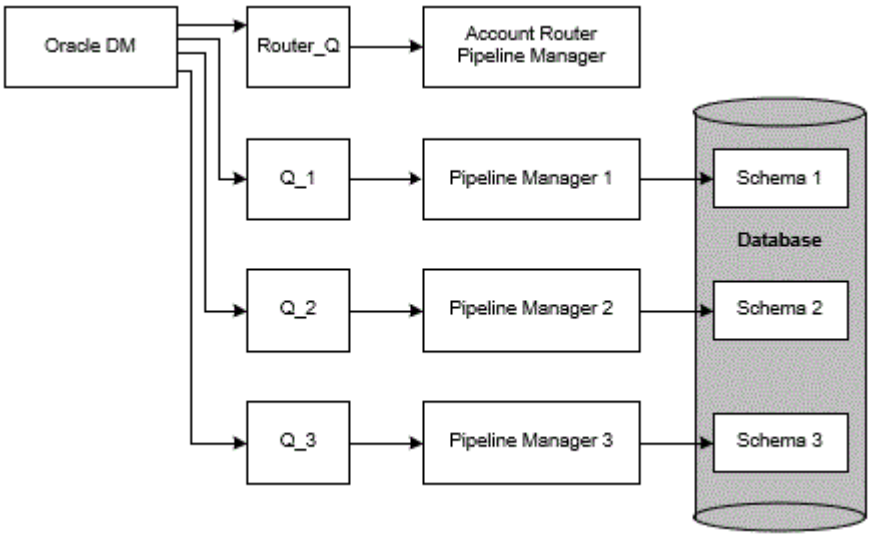
```
schema_name.queue_name
{
    Criteria
}
```

where:

- *schema_name* is the name of the database schema containing the queue. Include the schema name only if the queue resides in a remote database schema. Omit the schema name if the queue resides in the local database schema.
  In multischema systems, each instance of the Oracle DM connects to only one database queue in a schema. That schema is the DM's local schema. In some cases, the Oracle 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 **aq_event_map** file to send the events to the appropriate queue.
- *queue_name* is the name you assigned to the AQ database queue.
- *criteria* specifies the events to send to the AQ database queue. You can configure the Oracle 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 6-5.

Figure 6-5 Sample Synchronization Architecture



In this system, you must configure the **aq_event_map** file so that the Oracle DM 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 **aq_event_map** file requires the following additional entries:

```
ROUTER_Q
{
    HoldCDRProcessing
    ResumeCDRProcessing
    MigrateAcct
}

Q_1 # local database schema queue
{
    MigrateSource
    MigrateDestination
}
Q_2
{
    MigrateSource
    MigrateDestination
}
Q_3
{
    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 AQ database queue. To do this, connect the DAT_Listener modules to your queues by setting the following registry entries:

- Use the **InfranetConnection** entry to specify how to connect to the database schema that contains the queue. This entry points to the **LoginInfranet** registry section.
- Use the **QueueName** entry to specify the name of the 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 Linux 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 non-default 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".

7

Pipeline Manager Error Messages

Learn about the Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager error messages.

Topics in this document:

- Pipeline Framework Error Messages
- Pipeline Manager Module Error Messages
- Pipeline Utility 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 7-1 lists the Pipeline Framework error messages.

Table 7-1 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	Cannot 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*'.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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.
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 does not have 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*

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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*.
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	Cannot insert line *value* from table *value* into prefix description table.
ERR_DAT_PREFDESC_INS_TREE_FILE	Cannot insert line *value* from file *value* into prefix description table.
ERR_DATA_INVALID	The data in field *value* is 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*'.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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**.
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* exists more than one time.
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* could not be found.
ERR_EDR_BLOCK_NOT_FOUND	The specified block alias *value* could not be found.
ERR_EDR_BUILD_RECORD_NOT_FILLED	'*value*' - EDR build 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*'.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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*'.
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 is not found 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*'.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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.
ERR_INVALID_DATE	Cannot 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.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_LEN_IS_MISSING	The first item in field definition *value* must be a number.
ERR_LINE_NOT_IDENTIFIED	The line could not be identified: *value*
ERR_LINE_NOT_INSERTED_DOUBLE	Could not insert line into message DB (double key). Line *value*
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	Cannot get the factory to create EDRs in *value*.
ERR_NO_EVENTHANDLER_FOUND	Event handler not found in module '*value*'.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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.
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 exist 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*.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
ERR_PCM_ERROR	PCM Error: err: *value* field: *value* loc *value* errclass: *value* rec_id: *value* resvd: *value* resvd2: *value- value*
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	Cannot 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	Cannot 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*' cannot be opened.
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 be copied to *value* for test recycling.
ERR_REDO_POOL_ENTRY_NOT_FOUND	Redo pool entry '*value*' not found in function pool.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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.
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 too 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.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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*
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*'.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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.
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 is 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.

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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).
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

Table 7-1 (Cont.) Pipeline Framework Error Messages

Error Message	Description
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
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 7-2 lists the DAT_AccountBatch error messages.

DAT_AccountBatch

Table 7-2 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 exist: *value*.

Table 7-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_ACCOUNTBUSY_JOB_ALREADY_REMOVED	Busy job with the ID does not exist : *value*
ERR_ACCOUNTBUSY_JOB_ID_FIELD_NOT_FOUND	Busy job field not found in flist: *value*.
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 the same or different transaction, so not initializing the validity with current time.
ERR_GET_RANGE_ITEMS	DAT_LoginDbObject::getPoidRangeItems failure: Reason=*value*

Table 7-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_INIT_ACCOUNTS_CANCELLED	Thread=*value* has canceled in DAT::InitCustomerThread::initAccounts method - *value*
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*

Table 7-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_INSERTING_CUST_CREATE_EVENT_ORDER_PROFILE	Unable to insert EventOrderProfile (*value*) for newly created account (*value*) during a CustCreate event.
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*

Table 7-2 (Cont.) DAT_AccountBatch Error Messages

Error Message	Description
ERR_SCRATCH_PAD_ITEM_NULL	Attempting to call AdoptAndLock() will a NULL ScratchPadItem. Pipeline Transaction Hash=*value*
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 7-3 lists the DAT_AccountRealtime error messages.

Table 7-3 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 7-4 lists the DAT_BalanceBatch error messages.

Table 7-4 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*'.

Table 7-4 (Cont.) DAT_BalanceBatch Error Messages

Error Message	Description
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*
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 7-5 lists the DAT_BalanceRealtime error messages.

Table 7-5 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 7-6 lists DAT_ConnectionManager error messages.

Table 7-6 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*)

Table 7-6 (Cont.) DAT_ConnectionManager Error Messages

Error Message	Description
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*)
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 7-7 lists the DAT_ConnectionPool error messages.

Table 7-7 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 user ID (*value*)
ERR_LOGOUT_TO_CM	Logout from CM failed for user ID (*value*)
ERR_OPCODECALL_FAILED	Opcode call failed (*value*).
ERR_SYSTEM_ERROR	Unexpected error (*value*).

DAT_Currency

Table 7-8 lists the DAT_Currency error message.

Table 7-8 DAT_Currency Error Messages

Error Message	Description
ERR_REGULAR_EXP	Error in Regular Expression Compilation, Desc : *value*.

DAT_Discount

Table 7-9 lists the DAT_Discount error messages.

Table 7-9 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 7-10 lists the DAT_ExchangeRate error message.

Table 7-10 DAT_ExchangeRate Error Message

Error Message	Description
ERR_DAT_EXCHANGERATE_INS_LIST_DB	Error in line *'value'* in database table *'value'*.

DAT_InterConnect

Table 7-11 lists the DAT_InterConnect error messages.

Table 7-11 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*.
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 7-12 lists the DAT_ItemAssign error messages.

Table 7-12 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 7-13 lists the DAT_Listener error messages.

Table 7-13 DAT_Listener Error Messages

Error Message	Description
ERR_CONVERTING_FLIST	FLIST string cannot be converted. *'value'*.

Table 7-13 (Cont.) DAT_Listener Error Messages

Error Message	Description
ERR_CONVERTING_FLIST_TO_STR	Compact FLIST cannot be converted to string. *'value'*.
ERR_OPENING_QUEUE	Error : Could not open the Queue errorCode:*value*
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 7-14 lists the DAT_ModelSelector error messages.

Table 7-14 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*'.

Table 7-14 (Cont.) DAT_ModelSelector Error Messages

Error Message	Description
ERR_MODEL_SELECTOR_DUPLICATE	Cannot insert new model selector '*value*'.
ERR_MODELSELECTOR_LISTENER_NOT_FOUND	Listener '*value*' not found
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 7-15 lists the DAT_NumberPortability error messages.

Table 7-15 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 7-16 lists the DAT_PortalConfig error messages.

Table 7-16 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 7-17 lists the DAT_Price error messages.

Table 7-17 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 7-18 lists the DAT_Rateplan error messages.

Table 7-18 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 7-19 lists the DAT_Recycle error messages.

Table 7-19 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 7-20 lists the DAT_ResubmitBatch error messages.

Table 7-20 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 7-21 lists the DAT_ScenarioReader error messages.

Table 7-21 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 7-22 lists the DAT_USC_Map error messages.

Table 7-22 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 7-23 lists the DAT_Zone error messages.

Table 7-23 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 7-23 (Cont.) DAT_Zone Error Messages

Error Message	Description
WRN_DUPLICATE_SERVICETYPE	Duplicate entry for ServiceType=*value* corresponding to ServiceCode=*value*.

EXT_InEasyDB

Table 7-24 lists the EXT_InEasyDB error messages.

Table 7-24 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 7-25 lists the EXT_PipelineDispatcher error messages.

Table 7-25 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 7-26 lists the FCT_Account error messages.

Table 7-26 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 7-27 lists the FCT_AccountRouter error messages.

Table 7-27 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 7-28 lists the FCT_AggreGate error messages.

Table 7-28 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 7-29 lists the FCT_APN_Map error messages.

Table 7-29 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 7-30 lists the FCT_ApplyBalance error messages.

Table 7-30 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 7-31 lists the FCT_BatchSuspense error messages.

Table 7-31 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 7-32 lists the FCT_BillingRecord error messages.

Table 7-32 FCT_BillingRecord Error Messages

Error Message	Description
ERR_BALANCE_NOT_FOUND	Error adding discount balance info: *value*.

FCT_CallAssembling

Table 7-33 lists the FCT_CallAssembling error messages.

Table 7-33 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 7-33 (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 7-33 (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 7-34 lists the FCT_CarrierIcRating error messages.

Table 7-34 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 7-35 lists the FCT_CiberOcc error messages.

Table 7-35 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 7-36 lists the FCT_CliMapping error messages.

Table 7-36 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 7-37 lists the FCT_CreditLimitCheck error messages.

Table 7-37 FCT_CreditLimitCheck Error Messages

Error Message	Description
WRN_NO_BG_OBJECT	No Balance Group Object found

FCT_CustomerRating

Table 7-38 lists the FCT_CustomerRating error messages.

Table 7-38 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 7-39 lists the FCT_DataDump error messages.

Table 7-39 FCT_DataDump Error Messages

Error Message	Description
ERR_INSERT_EVENT	Failed to insert event *value*

FCT_Discard

Table 7-40 lists the FCT_Discard error messages.

Table 7-40 FCT_Discard Error Messages

Error Message	Description
WRN_FIELD_NOT_FOUND	EDR field *'value'* does not exist.

FCT_Discount

Table 7-41 lists the FCT_Discount error messages.

Table 7-41 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 7-41 (Cont.) FCT_Discount Error Messages

Error Message	Description
ERR_ROLLBACK_EDR	EDR transaction rollback demanded.

FCT_DroppedCall

Table 7-42 lists the FCT_DroppedCall error messages.

Table 7-42 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 7-43 lists the FCT_DuplicateCheck error messages.

Table 7-43 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 7-43 (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 7-44 lists the FCT_EnhancedSplitting error messages.

Table 7-44 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 7-45 lists the FCT_ExchangeRate error messages.

Table 7-45 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 7-46 lists the FCT_Filter_Set error messages.

Table 7-46 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 7-47 lists the FCT_IRules error messages.

Table 7-47 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 7-48 lists the FCT_IScriptPlugin error messages.

Table 7-48 FCT_IScriptPlugin Error Messages

Error Message	Description
ERR_SCRIPT_NOT_USABLE	The iScript *'value'* is not usable.

FCT_ItemAssign

Table 7-49 lists the FCT_ItemAssign error messages.

Table 7-49 FCT_ItemAssign Error Messages

Error Message	Description
ERR_INVALID_ITEM_POID	Invalid item poid returned from DAT_ItemAssign.

FCT_MainRating

Table 7-50 lists the FCT_MainRating error messages.

Table 7-50 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 7-51 lists the FCT_Opcode error messages.

Table 7-51 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 7-52 lists the FCT_PreRating error messages.

Table 7-52 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 7-53 lists the FCT_PreSuspense error messages.

Table 7-53 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 7-54 lists the FCT_RateAdjust error message.

Table 7-54 FCT_RateAdjust Error Message

Error Message	Description
ERR_ADD_RATEADJUST	Failure while adding rate adjust entry (*value*).

FCT_Reject

Table 7-55 lists the FCT_Reject error message.

Table 7-55 FCT_Reject Error Message

Error Message	Description
ERR_REJECT_UNKNOWN_STREAM	Error : Stream *'value'* is unknown.

FCT_Rounding

Table 7-56 lists the FCT_Rounding error message.

Table 7-56 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 7-57 lists the FCT_SegZoneNoCust error message.

Table 7-57 FCT_SegZoneNoCust Error Message

Error Message	Description
ERR_INIT_SEG_ZONE_LINK	Failure during initialization of segment zone link table.

FCT_Suspense

Table 7-58 lists the FCT_Suspense error messages.

Table 7-58 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 7-59 lists the FCT_Timer error messages.

Table 7-59 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 7-60 lists the FCT_TriggerBill error message.

Table 7-60 FCT_TriggerBill Error Message

Error Message	Description
ERR_TRIGGER_BILLING	TriggerBilling is required.

FCT_UoM_Map

Table 7-61 lists the FCT_UoM_Map error messages.

Table 7-61 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 7-62 lists the FCT_UsageClassMap error message.

Table 7-62 FCT_UsageClassMap Error Message

Error Message	Description
ERR_UCENTRY_INVALID	Failed to setup usage class mapping entry (*value*).

FCT_USC_Map

Table 7-63 lists the FCT_USC_Map error messages.

Table 7-63 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 7-64 lists the INP_GenericStream error message.

Table 7-64 INP_GenericStream Error Message

Error Message	Description
ERR_INP_GENERICSTREAM_PARSE_ERROR_STREAM	Parse error on input stream: *value*.

INP_Realtime

Table 7-65 lists the INP_Realtime error messages.

Table 7-65 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 7-65 (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 7-66 lists the INP_Recycle error messages.

Table 7-66 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 7-67 lists the NET_EM error messages.

Table 7-67 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	Linux 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 7-68 lists the OUT_DB error messages.

Table 7-68 OUT_DB Error Messages

Error Message	Description
ERR_DB_ROLLBACK_TRANSACTION	Database transaction rollback failed for stream *'value'*.
ERR_DB_STREAM_CLOSE	Database could not be 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 7-69 lists the OUT_GenericStream error messages.

Table 7-69 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 7-70 lists the OUT_Realtime error messages.

Table 7-70 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

Table 7-71 lists the LoadIFWConfig error messages.

Table 7-71 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*

8
Testing Pipeline Manager

Learn how to start, stop, and test the configuration of Oracle Communications Billing and Revenue Management (BRM) Pipeline Manager.

Topics in this document:

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.

See "Starting and Stopping Pipeline Manager"for more information about starting Pipeline Manager.

> **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.

See "Starting and Stopping Pipeline Manager" for more information about stopping Pipeline Manager.

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 **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. 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 8-1 Sample Format

```
service-code;a-number;b-number;start-time;duration;vol-sent;vol-recieved;callclass;cell-id;apn
```

Table 8-1 describes CDR field formats and restrictions.

Table 8-1 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)

Table 8-1 (Cont.) CDR Field Formats

Field	Description and Format
apn	The access point name for the call. Maximum length: 64 characters. Sample use: Specifying the URL for GPRS

Example 8-2 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**.

For example, you might see an error like the following when you try to start Pipeline Manager:

```
start_ifw_realtime
** /u01/app/brm/BRM/ifw/bin/start_ifw: NOTE: using IFW regfile /BRM/ifw/conf/
wirelessRealtime.reg
** /u01/app/brm/BRM/ifw/bin/start_ifw: ERROR:
/BRM/ifw/conf/wirelessRealtime.reg.lock EXISTS; remove to restart
```

Solution: If the BRM framework has not been stopped correctly, a lock file already exists.

Delete the lock file *Pipeline_home***/conf/wirelessRealtime.reg.lock** and restart Pipeline Manager.

- **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.