Oracle® Communications Billing and Revenue Management

Managing Customers

Release 15.1
F93231-01
April 2025

ORACLE

Oracle Communications Billing and Revenue Management Managing Customers, Release 15.1

F93231-01

Copyright © 2017, 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

11 Setting up Welcome Messages to Customers

12 Sending Email to Customers Automatically

13 Sending Messages to Customers through External Notification Applications

20 Managing Account and Bill Unit Hierarchies

21 Managing Charge and Discount Sharing Groups

31 Setting Up Customer Self Care with Self-Care Manager

Part IX Customer Management Utilities

32 Customer Management Utilities

Preface

This guide describes how to manage customers in Oracle Communications Billing and Revenue Management (BRM).

Audience

This guide is intended for operations personnel, developers, and system administrators.

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.

Part I

Managing Account Information

This part provides information about how to configure and manage your customers' information in Oracle Communications Billing and Revenue Management (BRM) accounts. It contains the following chapters:

1

Choosing Account Creation Options

Learn how to set options for how customer accounts are created in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Customer Information Collected When an Account is Created

When you create an account, you collect information about the customer such as:

- The package that the customer signs up for.
- Contact information, such as name, address, and telephone number.
- Service login and Web access names and passwords.
- Customer's language.
- Customer's account currency.
- Billing information, such as:
  - The accounting type (balance forward or open item).
  - The frequency of the billing cycle.
  - The payment method (credit card, invoice, and so on).
  - Payment information, such as credit card number or invoice billing address.

The information required by default is the last name, address, city, and country. You can change which information is required by using the Field Validation Editor.

If you create accounts by using a custom application or over the Web, you can create your own account creation fields. To do so, you need to create new fields in Storable Class Editor. The information you collect is stored in account profiles.

Specifying How to Validate Customer Contact Information

You can define the valid formats for your customers' contact information by using the Field Validation Editor. For example, you can require phone numbers to contain 10 digits and a dash (-) character.

You can also specify whether the information is required, whether it's case sensitive, and the minimum and maximum values (for example, the maximum length of a login name).

You can configure this information:

- **Login name format.** The default requirement is at least 1 character but no more than 255.
- **Password format.** The default requirement is at least 1 character but no more than 255.
- **Telephone number format.** BRM includes several standard formats, but you can add your own formats.
- **US state abbreviation.** The default required format is two uppercase letters such as CA.
- **Salutation.** For example, Mr. or Ms.
- **US ZIP code format.** Tax calculation programs validate ZIP codes. If you use a tax calculation package, make sure that your tax calculation program can handle 9-digit ZIP codes.

Specifying Defaults for Account Currency, Country, and Email

You can set up these systemwide defaults that are used by all account creation methods:

- Default account currency
- Default country
- Default customer email domain name

To specify these defaults:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Edit these entries:
   - To specify the default account currency, enter a currency code in the **currency** entry. For example, to make the Canadian Dollar the default account currency:

     ```
     - fm_cust_pol currency   124
     ```

     For a list of currency codes, see *BRM_home*/**include/pin_currency.h**.
   - To specify the default country, change the value of the **country** entry. For example, to make America the default country:

     ```
     - fm_cust_pol   country   USA
     ```

     The country name must conform to the validation rules for country names set in the Field Validation Editor. See "Specifying How to Validate Customer Contact Information".
   - To specify the default email domain, change the value of the **domain** entry. For example:

     ```
     - fm_cust_pol   domain   isp.nz
     ```
3. Save the file.

The new values become effective immediately and apply to the next account created. You don't need to restart the CM.

Allowing Accounts to Be Created with Backdated Services or Balances

By default, when you create an account, its associated services and balances must have a creation date on or after the account creation date. You can configure BRM to allow users to create accounts with the service or balance date backdated prior to the account creation date.

To enable this feature, run the **pin_bus_params** utility to change the **SubsDis74BackDateValidations** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To allow accounts to be created with backdated services or balances:

1. Go to *BRM_home*/**sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

3. In the file, set the **SubsDis74BackDateValidations** element to **enabled**:

   ```
   <SubsDis74BackDateValidations>enabled</SubsDis74BackDateValidations>
   ```

4. Save the file as **bus_params_subscription.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.

Customizing Account Creation

You can customize account creation in the following ways:

- To customize how account names are displayed, use the PCM_OP_CUST_POL_PREP_ACCTINFO policy opcode. For example, you can remove spaces from the name or capitalize the first and last name. The default is to store the name as it was entered at account creation.
- To change the format of account numbers, use the PCM_OP_CUST_POL_PREP_ACCTINFO policy opcode.

  By default, the account number is derived automatically from the account object POID. For example, if the POID is **0.0.0.1 /account 12225 19**, the account number is **0.0.0.1-12225**.

  The account number does not need to include any part of the POID; you can define your own format.
- To change the date format, use the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode.

  By default, BRM stores expiration dates as four-character strings in the format *MMYY*. The years from 00 and beyond are understood as the 2000 millennium. Valid formats are *MMYY, MM/YY, M/YY,* and *MM/YYYY*, all of which are converted to *MMYY*.
- To check for duplicate account creations, use the customer policy opcodes. For example, you can customize your customer validation policies to check for a duplicate name or credit card number. This allows you to prevent customers from accidentally signing up and getting billed for accounts they do not want. This might happen when a customer loses a

connection right after creating an account. Being unsure if the account was created, the customer creates a duplicate account.

- To validate account creation numbers, use the PCM_OP_CUST_POL_VALID_AACINFO policy opcode. When creating accounts, you can offer different packages to different customers by having customers enter account creation numbers. You can also assign account creation numbers to customers to track marketing information.
- To return a point-of-presence (POP) list, use the PCM_OP_CUST_POL_GET_POPLIST policy opcode.
- To prepare automatic account creation (AAC) data for validation, use the PCM_OP_CUST_POL_PREP_AACINFO policy opcode.
- To export information to a non-BRM database, use the PCM_OP_CUST_POL_POST_COMMIT policy opcode.
- To customize how to select schemas during account creation in a multischema system, use the PCM_OP_CUST_POL_GET_DB_LIST and PCM_OP_CUST_POL_GET_DB_NO opcodes.
- To store information collected when customers create an account by using a website, use the PCM_OP_CUST_POL_PREP_AACINFO policy opcode.

For more information, see "Creating Accounts" in *BRM Opcode Guide*.

2

Changing a Customer's Billing Configuration

Learn how to manage changes to customer billing information, such as changes to billing dates, credit limits, and payment methods.

Topics in this document:

Changing a Customer's Billing Day of Month

Use Billing Care or Customer Center to change the billing day of month.

A change to a billing day of month (DOM) takes effect in the next billing cycle. This means that there will be a partial billing cycle to handle the difference in days between the end of the current billing cycle and the start of the new billing cycle. For more information, see "Specifying How to Handle Partial Accounting Cycles" in *BRM Configuring and Running Billing*.

When you change the billing date and create a partial billing cycle, BRM prorates all cycle fees for that cycle.

Changing a Customer's Accounting Type

A bill unit's accounting type can be changed at any time. However, BRM does not validate the change nor take any actions other than changing the accounting type in the next billing cycle. You must ensure that the impact of any accounting type changes do not confuse your customers. For example:

- If the accounting type is changed from open item accounting to balance forward accounting, the customer's next bill would include all open and unpaid items. Your customer should be informed that the bill now includes any past due charges from previous billing cycles.
- If the accounting type is changed from balance forward accounting to open item accounting, the customer's next bill would not include any unpaid items. Your customer should be informed that charges from previous bills are still past due, even though they do not appear on the current bill.

Changing an Account's Bill Unit to Nonpaying

If you change a child account bill unit's payment method to nonpaying after creating the account, the billing information for the nonpaying bill unit changes to match the parent account. A nonpaying bill unit must use the same currency, billing date, billing frequency, and accounting type as its parent account. If the accounts use two currencies, the parent account and the nonpaying bill unit in the child account must use the same primary currency.

Changing a Closed Child Account's Bill Unit to Nonpaying

You can change a closed child account bill unit's payment method to nonpaying (subordinate) if either of the following conditions is true:

- The closed child account was billed for all the previous cycles.
- The closed child account was not billed because you have disabled billing of closed accounts and the child account's total balance due for the bill unit is zero. See "Suspending Billing of Closed Accounts" in *BRM Configuring and Running Billing* for more information about disabling the billing of closed accounts.

3

Collecting Information About Customers

Learn how to collect and manage information about your customers in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Making Notes About Customers

You can add notes to accounts to record information obtained from customers and to explain why you performed various actions.

Both Billing Care and Customer Center categorize notes to differentiate them when they are displayed in the Notes dialog box and to enable other tools, such as reporting applications, to gather a specific type of note from your BRM database. Note categories include:

- General Notes
- A/R Charge Credit Card
- A/R Credit Account
- A/R Credit Item
- A/R Debit Account
- A/R Open Dispute
- A/R Set Credit Limit
- A/R Settle Dispute
- A/R Write-Off
- Account/Service Status Change

General notes, such as notes about the best time of day to call a customer, apply to the entire account. Notes in all other categories apply to specific actions, such as adjustments, charges, credits, disputes, write-offs, and status changes.

A CSR can retrieve a list of notes associated with an account. Each note includes the name of the CSR who created.

Collecting a Summary of Activities Performed on an Account

You collect a summary of activities performed on an account by using Newsfeed. Enable Newsfeed for each event you want to collect data about.

You can enable Newsfeed for the following events:

- **Accounts receivable (A/R)**, which includes currency adjustments, noncurrency adjustments, open disputes, closed disputes, refunds, write-offs, write-off reversals, and collections.
- **Payments**, which includes payments, payment method changes, and payment reversals.
- **Account**, which includes name or contact information changes, account status changes, service status changes, and services (for example, a SIM card) attached or detached from the account, and billing information changes.
- **Charges**, which includes one-time charges, recurring charges, cycle charges, purchase charges, purchase and cancellation of a package or bundle, bill issued, and bill issued mid-cycle.

To enable Newsfeed for an event:

1. Open the *BRM_home***/sys/data/config/pin_notify** file in a text editor.
2. Uncomment the entry for the events for which you want to enable Newsfeed:

   ```
   180  0   event_name
   ```

   where *event_name* is the name of the event that you want a summary of.

   For example:

   ```
   180  0    /event/billing/account/adjustment
   ```

   This example means that opcode 180 (PCM_OP_ACT_PROCESS_EVENTS) is called when the **/event/billing/account/adjustment** event occurs.
3. Save and close the file.
4. Run the following command, which loads the event notification file into the BRM database:

   ```
   load_pin_notify   pin_notify_file
   ```
5. Stop and restart the Connection Manager (CM).

Customizing and Localizing Newsfeed

The strings for Newsfeed contain the basic text of a message and include the type of activity performed, such as account, dispute, payment, and so on.

You can customize and localize the strings for Newsfeed; for example “Payment reversed". To do so, you edit a copy of the **newsfeed.en_US** sample file in the *BRM_home***/sys/msgs/newsfeed** directory. If you are loading a localized version of this file, use the correct file extension for your locale. You then use the **load_localized_strings** utility to load the contents of the file into the **/strings** objects.

When you run the **load_localized_strings** utility, use this command:

```
load_localized_strings newsfeed.locale
```

For information on loading the **newsfeed.***locale* file, see "Loading Localized or Customized Strings" in *BRM Developer's Guide*.

> **Note:**
>
> To customize the values displayed in the news feed or to add new events for the news feed, use the PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode. For information, see "Customizing News Feed Values and Events" in *BRM Opcode Guide*.

To load the customized strings for Newsfeed:

1. Open the *BRM_home***/sys/msgs/newsfeed/newsfeed.***locale* file in a text editor.
2. If you want to add new strings for Newsfeed, add the strings and map them to the reason codes from a list of reasons appropriate to the event type that are defined in the **newsfeed.***locale* configuration file.

   For example:

   ```
   DOMAIN = "newsfeed"
   STR
           ID = 132 ;
          VERSION = 60 ;
           STRING = "Payment reversed" ;
   END
   STR
           ID = 133 ;
           VERSION = 60 ;
           STRING = "%s original payment, %s" ;
   END
   ```

   The PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode is used to customize values in News Feed and to add new events for News Feed. The placeholder character **%s** is replaced with the value supplied in the PIN_FLD_MESSAGE field in the input flist of PCM_OP_ACT_POL_PROCESS_EVENTS. See "Customizing News Feed Values and Events" in *BRM Opcode Guide*.
3. If you want to modify the existing strings for Newsfeed, search for the string and change the string value.
4. Save and close the file.
5. Load the strings using the **load_localized_strings** utility.

Customizing the List of Reasons for Account Changes

When making changes to accounts in Billing Care or Customer Center, CSRs can choose from a list of reasons that specify why a change was made.

By default, BRM provides reasons for changing an account such as:

- Activating or inactivating an account
- Closing or writing off an account
- Crediting, debiting, or adjusting an account
- Updating the payment method
- Updating tax information
- Charging an account

You can add, modify, or delete reasons by editing the **reasons.en_US** sample file in the *BRM_home*/**sys/msgs/reasoncodes** directory, where *BRM_home* is the directory in which the BRM server software is installed. You then use the **load_localized_strings** utility to load the contents of the file into **/strings** objects.

To run the **load_localized_strings** utility, use the following command:

```
load_localized_strings reasons.locale
```

where *locale* is the file extension (for example, **en_US**). If you are loading a localized version of this file, use the correct file extension for your locale.

For more information about the utility, see "load_localized_strings" in *BRM Developer's Guide*.

Maintaining an Audit Trail of BRM Activity

BRM provides support for auditing any object in the BRM database so that a record is kept of every version of the object for future reference. This can be used to track changes to customer profiles, customer payment information, and so on. An audit trail can also be used to track internal changes, such as changes to your price list.

> **Note:**
>
> Enabling an audit trail decreases system performance significantly. Keep an audit trail only for the BRM activity you feel is absolutely necessary.
>
> If an auditable field is updated multiple times within a transaction, it will record the latest value. To generate audit for each operation, the operations should be done in separate transactions.

To access audit trail information for a customer, you must write an application. Your application would take the pertinent customer account information (such as the customer account number and the general time the audit occurred) and find and retrieve the requested audit trail data.

See "Auditing Customer Data" in *BRM Developer's Guide* for the following:

- Instructions on accessing audit trail information from the BRM database.
- Information on how audit trails work.
- Instructions on making new and existing object fields auditable.

4

Managing Credit Cards During Account Creation

Learn how to manage credit cards during account creation in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Charging Customers at Account Creation

By default, BRM charges for purchase fees and the first cycle forward fees at account creation. When a customer creates an account, BRM uses the credit card processor to authorize the payment for the fees. The next time you run billing, the **pin_deposit_cc** billing utility deposits the credit card payment. (You should run the **pin_deposit_cc** utility as part of the **pin_bill_day** script. See "About the Billing Utilities" in *BRM Configuring and Running Billing*).

The total due amount for the account is charged immediately and the payment is allocated immediately to all open bill items. Therefore, after the account is created, it will have no pending amount due and no payments that have not been applied. You can use the **pin_bus_params** utility to configure BRM not to collect the credit card payments immediately For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To determine how to charge customers at account creation:

1. Go to *BRM_home***/sys/data/config**.
2. Use the following command to create an editable XML file from the accounts receivable instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

   This command creates an XML file named **bus_params_AR.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_AR.xml.out**, set **CCcollect** to **enabled** or **disabled**:

   ```
   <CCcollect>value</CCcollect>
   ```

   where *value* is:

   - **Enabled** to enable credit card collection. This is the default.
   - **Disabled** to disable credit card collection.

> **Caution:**
>
> BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. Use care when updating parameters in the file.

4. Save and exit the file.
5. Rename the **bus_params_AR.xml.out** file to **bus_params_AR.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_AR.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide.* For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

The new value is effective immediately. You do not need to restart the CM.

Which Fees Are Charged at Account Creation

Only fees for charge offers that are purchased as part of the account creation process are charged and billed immediately. All other charge offer fees, even for those charge offers purchased the same day as account creation, are charged in the first billing cycle. For example, if a customer creates an account and then purchases an additional charge offer an hour later, the additional purchase fee is charged at the end of the current billing cycle.

How General Ledger Revenue Is Reported for Account Creation Charges

Even though charges collected at account creation have been paid, the BRM general ledger (G/L) interface categorizes those charges as unbilled revenue until the first billing cycle.

Validating Credit Cards at Account Creation

By default, BRM validates credit cards by checking the customer's name and address and the credit card number. You might want to turn validation off if the connection to the credit card processor is offline.

A customer can create an account even when:

- The credit card transaction processing service is unavailable.
- The ZIP code is valid but the street address is wrong.

By default, a customer cannot create an account if:

- The wrong ZIP code was entered.
- The credit card is not valid.
- The address was not entered or could not be read by the credit card processor.

> **Note:**
>
> If an account is created without validating the credit card, the account will not be charged for any fees when it is created.

To enable or disable credit card validation, do the following:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Change the value of the **cc_validate** entry:
   - To enable credit card validation, enter **1**.
   - To disable credit card validation, enter **0**.
3. Save the file.

You do not need to restart the CM to enable this entry.

Avoiding Credit Card Revalidation

You can specify the credit card revalidation interval. For example, you can allow an interval of one hour before a credit card needs to be revalidated. This reduces the cost of credit card validations and allows customers who use the same credit card to create multiple accounts without having to validate the credit card more than once.

To specify the credit card revalidation interval:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the accounts receivable instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

   This command creates an XML file named **bus_params_AR.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_AR.xml.out**, set **CCRevalidationInterval** to the number of seconds applicable:

   ```
   <CCRevalidationInterval>value</CCRevalidationInterval >
   ```

   The default value is **3600** seconds (one hour).

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM configuration.

4. Save and exit the file.
5. Rename the **bus_params_AR.xml.out** file to **bus_params_AR.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

```
pin_bus_params bus_params_AR.xml
```

You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

The new value is effective immediately. You do not need to restart the CM.

Allowing Account Creation Without a Credit Card

You can allow your customers to create accounts and sign up for services without providing a credit card number when they create an account. This means that you can offer your customers a number of free days of service before they have to provide a credit card number.

To implement this feature, create the accounts by using the Undefined payment method. You can change the payment method to credit card or invoice after the free period has ended. You can also use the PCM_OP_CUST_SET_BILLINFO opcode to set the payment method. See *BRM Opcode Guide*.

Increasing Account Creation Speed When Paymentech Is Offline

If you know that your connection to Paymentech will be offline, you can speed up account creation by not allowing timeouts. Instead, a connection to Paymentech results in a “no answer" error immediately. By default, account creation can occur even though there is a “no answer" error. Also, a “no answer" error does not create checkpoint records, so you do not have to resolve the transaction.

> **Note:**
>
> - If you use this option, you cannot process credit card transactions by using the **pin_collect** or **pin_deposit** utilities. You must wait and run billing when the Paymentech connection is restored.
> - If the credit card payment service is not available and you still want to create accounts, you must isolate those accounts for later credit card authorization. Modify the PCM_OP_PYMT_POL_VALIDATE policy source file either to save a list of permissive account creation or to send email to the system administrator. Alternatively, you can write a simple application to periodically check accounts and flag the accounts that have been created without verification.

1. Open the Paymentech DM configuration file (*BRM_home*/**sys/dm_fusa/pin.conf**).
2. Edit the **online_proto** entry:
   - Enter **linkdown** to disable timeouts and report “no answer" for all connections.
   - Enter **socket** to enable the connection to function normally.
3. Edit the **batch_proto** entry:

- Enter **linkdown** to disable timeouts and report “no answer" for all connections.
- Enter **socket** to enable the connection to function normally.

4. Save the file.
5. Stop and restart the Paymentech DM.

Specifying the Account That Records Credit Card Validations

To validate credit cards, BRM needs an account. By default, BRM logs credit card validations against the root account during account creation.

If you want to use a different account, do the following:

1. Open the Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**). *BRM_home* is the directory in which the BRM server software is installed.
2. Change the account number in the following entry:

   ```
   - fm_pymt_pol validate_acct database_number /account 1
   ```

   where *database_number* is the database number of the BRM database; by default, this is 0.0.0.1.
3. Save the file.

You do not need to restart the CM to enable this entry.

To customize this feature, use the PCM_OP_PYMT_POL_SPEC_VALIDATE policy opcode. See "Validating Credit Card and Direct Debit Transactions" in *BRM Opcode Guide*.

5

Managing Passwords and Login Names

Learn how to manage customer authentication, including login names, passwords, and security codes, in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Login Names and Passwords

By default, all services require a login name and password. For services such as telephony where a customer does not use a login and password, logins and passwords can be generated automatically for internal use.

- By default, the login name can be a minimum of 1 character and a maximum of 255 characters.
- By default, the password must be a minimum of 1 character and a maximum of 255 characters.
- Login names are unique and can be assigned to only one account.

You can change the minimum and maximum login name and password lengths by using the Field Validation Editor. See "Specifying How to Validate Customer Contact Information".

All login names and passwords are associated with a service, such as the broadband service (***/service/ip***). Customer service representatives (CSRs) use a login name and password to log in to the **admin_client** service.

BRM uses two types of customer passwords:

- Customers use *service passwords*, such as the password that a customer provides when logging in to a broadband connection, to access a broadband service.
- Customers use *account passwords* for non-IP access, such as accessing a Web page.

By default, account passwords are stored in the database in an encrypted format; service passwords are not.

To customize password encryption, use the PCM_OP_CUST_POL_ENCRYPT_PASSWD, PCM_OP_CUST_POL_COMPARE_PASSWD, and PCM_OP_CUST_POL_DECRYPT_PASSWD opcodes. See "Creating Passwords" in *BRM Opcode Guide*.

For more information about encryption, see "About Encrypting Data" in *BRM Developer's Guide*.

About Telco Service Logins and Passwords

When you create an account that uses telco services, the customer ID and password are generated automatically. Therefore, a CSR does not need to enter an ID and password at account creation or when adding a telco service.

> **Note:**
>
> Internally, the customer ID is the same as the login name.

- To ensure that a unique ID is generated, the default ID is a unique string composed of the following elements:
  - A timestamp generated by the Connection Manager (CM) that was used for creating the account.
  - The process ID (PID) of the CM.
  - The thread ID of the CM (always 1).
  - The CM host name.

  For example:

  269-20011128-095216-7-22493-1-*host_name*

  When an ID is needed: for example, for Web-based account management: the customer enters their MSISDN or IMSI. Applications can retrieve the MSISDN or IMSI from the customer's service objects. (Customers can also enter the ID.)

  > **Note:**
  >
  > When using an MSISDN or IMSI as a login, the customer must enter the full number with no punctuation, such as 014085551212.

  To customize how IDs are generated, you customize the PCM_OP_CUST_POL_PREP_LOGIN policy opcode.
- For security purposes, BRM generates a random eight-character password. You cannot change the password when a service is being added, but you can change it later.

  To customize how passwords are generated, you customize the PCM_OP_CUST_POL_PREP_PASSWD policy opcode.

Using Security Questions

You can specify two security questions for a customer. When a customer calls a CSR, the CSR asks the customer the security question, which is displayed in Customer Center or Billing Care.

Unlike service passwords, security codes are not validated by BRM; therefore, you cannot enforce properties such as the number of characters in a security code.

Assigning Passwords Automatically

You can set up account creation to do either of the following:

- Require the customer to specify a password. This is the default.
- Generate a password automatically for the customer.

To generate a password for the customer, you must supply the algorithm for generating passwords. To do so, customize the PCM_OP_CUST_POL_PREP_PASSWD policy opcode. See "Customizing Passwords" in *BRM Opcode Guide*.

Defining Email Login Names

> **Note:**
>
> Changing the customer's email login name also changes the customer's email address. Before changing a login name, make sure the customer wants to change the email address.

To change the requirements for email login names, use the PCM_OP_CUST_POL_PREP_LOGIN and PCM_OP_CUST_POL_VALID_LOGIN policy opcodes. See "Customizing Login Names" in *BRM Opcode Guide*.

The default email login requirements are:

- The email login must use all lowercase characters.
- The email login must include the domain, in this format:

  ```
  login@domain
  ```

  For example: francisco@example.com

Detecting Duplicate Logins

By default, BRM does not check for duplicate logins. This means that more than one customer can log in to a service by using the same name. To check for duplicate logins, use the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode. See "Enabling Duplicate Session Checking" in *BRM Opcode Guide*.

6

Managing Account and Service Status

Learn how to change account and service status in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

For basic information about account status, see "About Activating, Inactivating, and Closing Accounts" in *BRM Concepts*.

Automatically Inactivating and Closing Accounts

By default, BRM automatically inactivates accounts in the following situations:

- A credit card payment fails because a nonvalid credit card is used.
- A credit card validation fails, and the item is 30 or more days past due.
- The account is a child account with a nonpaying bill unit (**/billinfo** object), and the parent account is inactivated.

> **Note:**
>
> By default, account status is not changed when a credit limit is reached. However, also by default, service authorization requires that the credit limit has not been reached, so reaching a credit limit prevents a customer from using a service.

To change these defaults, you must customize the following policy opcodes:

- PCM_OP_PYMT_POL_COLLECT
- PCM_OP_ACT_POL_EVENT_LIMIT
- PCM_OP_ACT_POL_SPEC_VERIFY

> **Note:**
>
> When you *close* an account or service, you cannot schedule reactivation. Closed accounts and services must be reactivated manually.

See *BRM Opcode Guide* for more information.

Scheduling Status Changes in Advance

In Billing Care or Customer Center, you can schedule account and service status changes for a future date. You use a daily billing utility, **pin_deferred_act**, to change the status automatically on the scheduled date.

You can schedule the following types of actions to occur on a future date by:

- Changing the status of an account or service.
- Adding an account to a hierarchy.
- Removing an account from a hierarchy.
- Moving an account between hierarchies.

After an action is deferred, you can display, run, reschedule, and cancel the deferred action.

Scheduling account reactivation works differently for inactivating and closing:

- When you *inactivate* an account or service, you can set a future date to reactivate the account or service, or you can reactivate the account or service manually.
- When you *close* an account or service, you cannot schedule reactivation. Closed accounts and services must be reactivated manually.

Use the **pin_bill_day** script to run the **pin_deferred_act** utility daily. To ensure that the account status is correct before running billing, it is the first billing utility run by the **pin_bill_day** script. See "pin_deferred_act".

Changing Start Times for Reactivated Charge and Discount Offers

By default, when inactive charge and discount offers are reactivated, their purchase, usage, and cycle start times are changed to the current (reactivation) date. You can use the Connection Manager (CM) configuration file **change_start_time_on_activation** entry to change this behavior so that when inactive charge and discount offers are reactivated, the original purchase, usage, and cycle start times are kept.

This setting does not affect charge offers whose purchase is deferred to a later date. If the charge offer is reactivated before the purchase date, the original purchase, usage, and cycle start times are kept, even if this option is enabled.

> **Note:**
>
> The purchase and cycle start times determine the date on which the purchase and cycle charges begin to be applied to the charge and discount offers.

To keep the original purchase, usage, and cycle start times:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**), where *BRM_home* is the directory in which the BRM server software is installed.
2. Add the following entry:

```
-fm_bill change_start_time_on_activation value
```

where *value* is one of the following:

- **0** keeps the original purchase, usage, and cycle start times.
- **1** changes the purchase, usage, and cycle start times to the reactivation date. This is the default.

3. Save and close the file.
4. Stop and restart the CM.

Allowing Active Services with Inactive Accounts

By default, inactivating an account inactivates all services owned by the account. To allow inactive accounts to have active services, use the **allow_active_service_with_inactive_account** entry in the CM **pin.conf** file.

To allow active services with inactive accounts:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
2. Add the following entry:

```
-fm_cust allow_active_service_with_inactive_account value
```

where *value* is one of the following:

- **0** prohibits active services with inactive accounts. This is the default.
- **1** allows inactive accounts to have active services.

3. Save and close the file.
4. Stop and restart the CM.

Inactivating and Closing Accounts in Hierarchies

Changing the status of an account in an account hierarchy changes the status of all accounts inside and outside the account hierarchy that have at least one nonpaying bill unit whose paying parent bill unit is owned by the initially changed account.

For more information, see "How Account Status Changes Affect Hierarchies".

You cannot prevent the status of a *nonpaying* child bill unit from changing when its parent bill unit's status changes.

> **Note:**
>
> Changing the account status of a child account does not affect the status of its parent.

Reusing Login Names and Passwords from Closed Accounts or Canceled Services

By default, you cannot reuse the login name of a closed account or canceled service, although BRM has no restriction on reusing passwords.

To reuse login names, set up BRM to automatically modify the login name when you close an account. For example, you could have the string *#CLOSED#* prepended to the login name.

You can customize the PCM_OP_CUST_POL_PREP_STATUS policy opcode to automatically add characters to a login name when an account's status changes.

7

Backdating Subscription Actions

Learn how to backdate subscription actions in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Backdating Subscription Actions

Subscription actions backdating lets you perform certain subscription actions so that the operations take effect on a prior date instead of the date it occurred. You can perform subscription backdating operations to rectify errors committed while performing subscription operations.

For example, you might want to backdate a subscription operation when:

- A customer requests a charge offer cancellation on October 30, but the cancellation is not recorded in the BRM system until November 15. The customer is charged for the period of October 30 to November 15.
- A customer requests a charge offer purchase on January 1, but the purchase date is entered in the BRM system as February 1.

BRM supports the following subscription backdating operations:

- Backdated account creation.
- Backdated account and service status change. See "About Backdated Status Change".
- Backdated charge offer, discount offer, or bundle cancellation. See "About Backdated Charge Offer, Discount Offer, or Bundle Cancellation".
- Backdated charge offer or discount offer purchase, usage, or cycle start and end date.
- Backdated remaining charge discount offer purchase, cancellation, and modification *if* the proration options are set to **Prorated discount**. See the discussion of proration options in *BRM Creating Product Offerings*.

You can perform subscription backdating by using one of the following methods:

- By using Billing Care.
- By using Customer Center.
- By passing the backdated date in the PIN_FLD_END_T field of the corresponding subscription opcodes.

To backdate a charge offer, discount offer, bundle, or package purchase, use the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT opcode. See *BRM Opcode Guide*.

About Backdated Status Change

BRM supports backdating the status change of a charge offer, discount offer, service, or account to a prior date.

BRM does not allow backdating the status change in the following situations:

- The backdated date is prior to the G/L posting date. See "About Backdating Beyond the G/L Posting Date".
- The backdated date is prior to the effective date of the account, service, charge offer, or discount offer. See "How Effective Time Is Used to Validate Backdating Operations".
- The backdated date is prior to the date of the last status change. You can backdate any status change only up to the date the last status change happened; undoing previous status changes is not allowed.

  For example, if you change the status of an account from active to inactive on September 1, you cannot backdate a status change on October 1 to a date prior to September 1.

How Cycle Fees are Calculated for Backdated Status Change

When you backdate the status change, a charge or a refund is applied for the backdated period.

For example, consider that on July 1, you backdate the status of an active account to make it inactive effective from June 1. If the cycle fee is already applied for the period of June 1 to July 1, another cycle fee event is generated that calculates and refunds the charges.

You must rerate any usage events that have occurred in the backdated period to account for the status change.

About Backdating Beyond the G/L Posting Date

BRM does not allow backdating beyond the G/L posting date. When you post a G/L report, you prevent backdating adjustments, write-offs, or subscription transactions, prior to the last date you posted the G/L report. This maintains general ledger data integrity. After they are posted, the G/L report updates the **/data/ledger_report** object. This object records the date of the latest posted G/L report and controls transaction backdating.

You can run G/L reports at any time without posting data. When you do not post data, backdating is not affected.

How Effective Time Is Used to Validate Backdating Operations

BRM does not allow backdating if the date to which you want to backdate the operation on an account, service, charge offer, or discount offer is prior to the respective effective dates (creation dates).

Rerating of Events for the Backdated Period

Backdated purchase or cancellation of a charge offer or discount offer automatically triggers rerating. In these cases, BRM uses event notification to create rerate jobs that rerate the

events in the charge offer or discount offer. In all other subscription backdating scenarios, you must either run rerating manually or configure BRM to created rerate jobs automatically.

For example:

- You might manually need to rerate the usage events that had been rated prior to the backdating action.

  For example, on October 15, a usage event for a charge offer is charged at $2.00 per minute. On November 1, a new charge offer is purchased backdated to September 1. The new charge offer has usage pricing of $1.00 per minute and has a higher priority than the first charge offer. In order for the usage that occurred on October 15 to get rated with the new charge offer, you must rerate the usage events manually.

- If you backdate the cancellation of a shared discount offer, you will need to rerate the accounts or services in the group, because the accounts or services might have used the discount offer.

You can configure BRM to rerate backdated operations. See "Configuring Rerating" in *BRM Rerating Events*.

About Backdated Charge Offer, Discount Offer, or Bundle Cancellation

The cancellation of a charge offer can be backdated as far back as the purchase date of the charge offer.

> **Note:**
>
> - If you backdate a discount offer purchase to a date prior to the charge offer purchase date, the discount will not be applied in the bill of the current cycle. The discount will be applied in the bill of the next cycle.
> - If you backdate the purchase of a discount offer on a cycle forward or cycle arrears event to a previous accounting cycle, the discount offer will be refunded only for the current accounting cycle.

BRM does not allow you to backdate the charge offer, discount offer, or bundle cancellation if:

- The backdated cancellation date is prior to the G/L posting date. See "About Backdating Beyond the G/L Posting Date".
- The backdated cancellation date is prior to the charge offer or discount offer's effective date. For example, a charge offer or discount offer that was purchased to be effective from January 15 cannot have any event on it backdated prior to January 15. See "How Effective Time Is Used to Validate Backdating Operations".
- The backdated cancellation date is prior to the date of the last status change of the account or service.

  For example, consider that you create an account on December 1. On December 10, you change the status of the account to **Inactive**. On December 15, you change the status of the account back to **Active**. You cannot backdate a charge offer cancellation prior to December 15, which is the date of the last status change.

Note:

- When you backdate the cancellation of a fold, rollover charge offer, or billing time discount, you must run rerating to apply the corrections on the events that occurred after the backdated date.
- Backdating discounts with discount offer proration options is supported only for discounts that are prorated.

8

Managing Custom Service Life Cycles

Learn how to use custom life cycles to manage service status at a detailed level in Oracle Communications Billing and Revenue Management (BRM). For example, you can use statuses such as *Dormant*, *Credit Expired*, and *Fraud Investigated* to indicate the exact state of a service and control how the service is used.

Topics in this document:

About Custom Service Life Cycles

By default, all BRM service types use the following statuses: Active, Inactive, and Closed. If you need a life cycle that better represents the phases of a particular service type, create a custom service life cycle for that service type. For example, you might create the following custom life-cycle states:

- *Preactive* to indicate a service that is ready to activate, but not yet activated.
- *Fraud Investigated* to indicate a service that is under investigation.
- *Dormant* to indicate a service that hasn't been used for a specified amount of time.

Custom service life cycles can contain any number of states and state transitions. For each state, you can specify the following:

- The states to which the state can change
- The actions that occur before and after a state transition takes place
- Rules to validate requests received in the state

You can associate custom life cycles with any BRM service type.

To create custom life cycles, you edit and load the **config_lifecycle_states.xml** file. By default, the file contains a sample prepaid service life cycle, which you can use as is, modify, or delete. See "About the Sample Prepaid Service Life Cycle" for more information.

For the defined life cycle states for subscriber life cycle management, see "About the Sample Service State-to-Status Mapping".

> **Note:**
>
> The default service life cycle status is stored in the PIN_FLD_STATUS field in **/service** objects. The custom service life cycle state is stored in the PIN_FLD_LIFECYCLE_STATE field of **/service** objects.

Triggering State Changes in Custom Service Life Cycles

The state of a service changes as the result of actions or conditions that affect the service. The actions can be triggered manually or automatically.

In custom service life cycles, state changes can be triggered by the following:

- **CSRs:** Using Billing Care or Customer Center, a customer service representative (CSR) can manually perform any permitted state change. Such changes are defined in the **<TRANSITIONS>** child element of the **<STATES>** element in the **config_lifecycle_states.xml** file.
- **Any action that impacts a service balance:** After a balance is adjusted or topped up, the PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE policy opcode triggers any required service state changes and updates the state expiration date (PIN_FLD_SERVICE_STATE_EXPIRATION_T field) in the **/service** object based on the new state's expiration period.

  If you create your own custom service life cycles, you can modify this policy opcode to trigger state changes based on different criteria.
- **pin_state_change:** This utility performs bulk service state changes based on the state expiration time (PIN_FLD_SERVICE_STATE_EXPIRATION_T field in **/service** objects).

  A system administrator schedules the utility to run at a specified time each day. When the utility finds services whose state expires on the current date, it changes that state to its default next state.

  > **Note:**
  >
  > The default next state is specified in the **<TRANSITIONS>** element whose **<DEFAULT_FLAG>** is set to **1**. If this flag is not set to **1** in any of the transitions configured for a state, this utility does not change the state.

  See "pin_state_change" for more information.

Configuring Business Rules for Custom Service Life Cycles

For each life-cycle state, you can configure business rules to validate the actions that subscribers try to perform in the state.

You configure business rules in the **<RULES>** child element of the **<STATES>** element in the **config_lifecycle_states.xml** file. In that file, each business rule includes the following elements:

- **<MODULE>:** The name of the facilities module (FM) that uses the rule. (The rule is coded in the FM).

- **<NAME>:** The name of the rule as it appears in the FM.
- **<VALUE>:** A value that indicates whether the business rule is enabled or disabled for the state.

To create and use a custom business rule:

1. Code the validation logic for the new rule in the appropriate policy opcode.
2. Include a name for the rule in the policy opcode.
3. Configure a **<RULES>** element for the rule in the appropriate service life cycle state.

Additional Tasks for Creating Custom Service Life Cycles

In addition to defining and loading custom life cycles, you need to do the following:

1. Enable customer service life cycles in BRM. See "Enabling BRM to Use Custom Service Life Cycles".
2. Add Service Lifecycle Management (SLM) entries to the Connection Manager (CM) **pin.conf** file. See "Adding SLM Entries to the CM pin.conf File".
3. Map each state in the custom service life cycle to a status in the default service life cycle. See "Mapping States to Statuses".
4. Associate one or more services with the custom service life cycle. See "Associating Services with Custom Life Cycles".
5. Associate account bill units (**/billinfo** objects) with the SLM business profile. See "Associating Bill Units with the SLM Business Profile".

Creating Custom Service Life Cycles

To create a custom service life cycle:

1. Open the **config_lifecycle_states.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home***/sys/data/config** directory.

   > **Note:**
   >
   > The **config_lifecycle_states.xml** file can contain multiple service life cycle configurations. In the XML file, each lifecycle configuration is identified by its **<NAME>** element. When the content of the XML file is loaded into the BRM database, each lifecycle configuration is put into a separate **/config/lifecycle_states** object.

2. Add a **<ConfigObject>** element to the **<ObjectList>** element.
3. In the **<ConfigObject>** element, specify values for the lifecycle elements.
4. Add a **<STATES>** element to the **<ConfigObject>** element.
5. In the **<STATES>** element, specify values for the elements listed in table Table 8-1.

   For example:

```
<ObjectList
 xmlns=http://www.oracle.com/schemas/BusinessConfig
```

```
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation=http://www.oracle.com/schemas/BusinessConfig
__PIN_HOME__/xsd/config_lifecycle_states.xsd>

  <ConfigObject>
     <DESCR>Default Lifecycle State config Object</DESCR>
     <NAME>Default Lifecycle Configuration</NAME>
     <PERMITTED_SERVICE_TYPES elem="0">
         <SERVICE_TYPE>/service/telco/gsm/telephony</SERVICE_TYPE>
     </PERMITTED_SERVICE_TYPES>
     <STATES elem="0">
         <INITIAL_STATE>1</INITIAL_STATE>
         <SERVICE_STATE_EXPIRATION_T>0</SERVICE_STATE_EXPIRATION_T>
         <STATE_ID>101</STATE_ID>
         <STATE_NAME>Preactive</STATE_NAME>
         <STRING_ID>101</STRING_ID>
         <STR_VERSION>1</STR_VERSION>
         <RULES elem="0">
                   <MODULE>AAA</MODULE>
                   <NAME>REQ_ALLOWED</NAME>
                   <VALUE>Yes</VALUE>
         </RULES>
         <RULES elem="1">
                   <MODULE>AAA</MODULE>
                   <NAME>MO_ENABLED</NAME>
                   <VALUE>Yes</VALUE>
         </RULES>
         <RULES elem="2">
                   <MODULE>AAA</MODULE>
                   <NAME>MT_ENABLED</NAME>
                   <VALUE>No</VALUE>
         </RULES>
         <TRANSITIONS elem="0">
                   <DEFAULT_FLAG>1</DEFAULT_FLAG>
                   <NEXT_STATE>102</NEXT_STATE>
                   <POST_OPCODE>0</POST_OPCODE>
                   <PRE_OPCODE>0</PRE_OPCODE>
         </TRANSITIONS>
     </STATES>
```

Table 8-1 XML Elements Within the <STATES> Element

XML Element	Description	Notes
INITIAL_STATE	Integer that specifies whether this is the initial state of the life cycle.	Specify one of the following values: • **1:** Initial state. Only one state in the life cycle can have this value. • **0:** Not the initial state.

Table 8-1 (Cont.) XML Elements Within the <STATES> Element

XML Element	Description	Notes
SERVICE_STATE_EXPIRATION_T	Character string that specifies the number of days, hours, and minutes between the time the service changes to this state and the time it automatically changes to the default next state in its life cycle. **Note**: If **SERVICE_STATE_EXPIRATION_T** is set to **0**, the service remains in the same lifecycle state indefinitely, unless a manual transition is initiated. Set **SERVICE_STATE_EXPIRATION_T** to **0** for states with no default transition, otherwise an error will be returned. Even if you define the transition, but set **SERVICE_STATE_EXPIRATION_T** to **0**, the **pin_state_change** utility will skip the service. To skip the services with no default transitions when running the **pin_state_change** utility, set **SERVICE_STATE_EXPIRATION_T** to **0**. BRM uses the state's start date (PIN_FLD_LAST_STATUS_T field of the /**service** object) and the amount of time specified in this element to set the state's end date (PIN_FLD_STATE_EXPIRATION_T field of the /**service** object).	Specify a value in one of the following formats: • *days* • *days:hrs* • *days:hrs:mins* where: • *days* is the number of days. • *hrs* is the number of hours. • *mins* is the number of minutes. For example, 365:0:600 specifies 365 days, 0 hours, and 600 minutes. There are no minimum or maximum numbers for this value.
STATE_ID	Integer that identifies a state in a custom service life cycle.	Use 109 or greater for any custom states that you create. Each state ID must be unique. The sample prepaid service life cycle uses the following state IDs: • 101: Preactive • 102: Active • 103: Recharge Only • 104: Credit Expired • 105: Dormant • 106: Fraud Investigated • 107: Suspended • 108: Closed • 109: Suspended Active See "About the Sample Prepaid Service Life Cycle"
STATE_NAME	Character string used as the name of the state (for example, **Active**). This string is mapped to the PIN_FLD_STATE_NAME field in the **/config/lifecycle_states** object.	Minimum length is 1 character. Maximum length is 255 characters. **Note**: Values in this field are used to populate a list of service states in the **Change state to** field on the **Change Account/Service Status** panel in Customer Center. They also appear in the **Current state** field on that panel and in the **Status** column in the **Services** tab. When creating the string, take Customer Center UI length restrictions into consideration.
STRING_ID	The ID of the state.	None

Table 8-1 (Cont.) XML Elements Within the <STATES> Element

XML Element	Description	Notes
STR_VERSION	The version of the state.	None
RULES	Parent element of the elements that define a business rule for this state. This element is mapped to the PIN_FLD_RULES array in the **/config/lifecycle_states** object. The array contains all the business rules configured for this lifecycle state. The rules validate the actions that subscribers try to perform in the state. For example, the rules configured for the Recharge Only state in the sample prepaid service life cycle permit subscribers to receive calls and to use free aspects of the service, but they do not permit subscribers to make prepaid calls. See "Configuring Business Rules for Custom Service Life Cycles" and "About the Sample Business Rules" for more information.	If you configure multiple **<RULES>** elements, the integer value of each element's **elem** attribute must be different. The values do not have to be sequential, and they have no relationship to the values of the **elem** attributes of other elements.
MODULE	Character string that identifies the facilities module (FM) that validates this rule. For example, the AAA module validates the rules configured for the sample prepaid service life cycle.	Use any string that identifies the validating FM. This string is only informational. It does not appear in the code. Minimum length is 1 character. Maximum length is 255 characters.
NAME	Character string used as the name of the business rule (for example, **REQ_ALLOWED**). **Note:** The rule is coded in an FM.	Use a string that matches the name of the rule as it is coded in the FM. Minimum length is 1 character. Maximum length is 255 characters.
VALUE	Character string that specifies whether the rule is enabled or disabled.	Specify one of the following values: • **Yes**: Enabled • **No**: Disabled
TRANSITIONS	Parent element of the elements that define how this state changes to another state. This element is mapped to the PIN_FLD_TRANSITIONS array in the **/config/lifecycle_states** object. The array contains all the states to which this state can change. If a state is not listed in the array, the current state cannot be changed to it. For more information on how transitions are triggered, see "Triggering State Changes in Custom Service Life Cycles"	If you configure multiple **<TRANSITIONS>** elements, the integer value of each element's **elem** attribute must be different. The values do not have to be sequential, and they have no relationship to the values of the **elem** attributes of other elements.
DEFAULT_FLAG	Integer that specifies whether this is the default state for the status to which it is mapped. A status can be mapped to multiple states, but only one of those states can be the status's default state. If a state is required when only a status value is available, BRM uses the default state for that status.	Specify one of the following values: • **1**: Default state. If multiple states are mapped to a status, only one of those states can have this value. • **0**: Not the default state.
NEXT_STATE	Integer that identifies the state to change the service to.	Specify the state's unique ID.

Table 8-1 (Cont.) XML Elements Within the <STATES> Element

XML Element	Description	Notes
POST_OPCODE	Integer that identifies the number of the policy opcode you want BRM to call immediately after the state transition occurs. For example, you might add logic to the policy opcode to notify subscribers that their balance is insufficient and needs to be replenished.	Specify the opcode number of the policy opcode to run. Opcode numbers are listed in the **pcm_ops.h** file in the *BRM_home*/**include** directory. Enter **0** to call no opcode.
PRE_OPCODE	Integer that identifies the number of the policy opcode you want BRM to call immediately before the state transition occurs. For example, you might add logic to the policy opcode that notifies subscribers when their balance is insufficient and needs to be replenished.	Specify the opcode number of the policy opcode to run. Opcode numbers are listed in the **pcm_ops.h** file in the *BRM_home*/**include** directory. Enter **0** to call no opcode.

6. Save and close the file.
7. Open the *BRM_home*/**apps/load_config/pin.conf** file.
8. Add the following line to enable validating the customer life cycles:

   ```
   - load_config validation_module libLoadValidSLM LoadValidSLM_init
   ```

9. Save and close the **pin.conf** file.
10. Run the following command, which loads the **config_lifecycle_states.xml** file:

    ```
    load_config config_lifecycle_states.xml
    ```

    If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file.
11. Stop and restart the CM.

> **Note:**
>
> You must configure the custom lifecycle states in ECE to stay synchronized with the lifecycle states added in BRM. For more information, see "Modifying Custom Service Life Cycles in ECE" in *ECE Implementing Charging*.

Modifying Custom Service Life Cycles

To modify a custom service life cycle:

1. Open the **config_lifecycle_states.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home*/**sys/data/config** directory.
2. Modify the **<ConfigObject>** element in which the life cycle is configured.

> **Note:**
>
> If you change the **<NAME>** element of a life cycle that services are using, you must update the value of the **lifecycle_obj** key in those services' validation templates in the **pin_slm_business_profile.xml** file. You must then reload that file.
>
> For details about editing and loading **pin_slm_business_profile.xml**, see "Associating Services with Custom Life Cycles".

3. Set the **configMode** attribute of the **<ObjectList>** element to one of the following values:
   - **replace:** Updates the existing **/config/lifecycle_states** objects. This is the default.
   - **recreate:** Deletes the existing **/config/lifecycle_states** objects and creates new objects.
4. Save and close the file.
5. Run the following command, which loads the modified **config_lifecycle_states.xml** file:

   ```
   load_config config_lifecycle_states.xml
   ```

> **Note:**
>
> - The "load_config" utility needs a configuration (**pin.conf**) file in the directory from which you run the utility. The **pin.conf** file must also contain the following entry, which enables the utility to validate the XML file values.
>
>   ```
>   - load_config validation_module libLoadValidSLM LoadValidSLM_init
>   ```
>
> - If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file.

   The utility converts the XML file into one or more **/config/lifecycle_states** objects, depending on the number of **<ConfigObject>** elements in the XML file. Each object contains the life cycle defined in one **<ConfigObject>** element.
6. Stop and restart the CM.

Enabling BRM to Use Custom Service Life Cycles

To use custom service life cycles, you must run the **pin_bus_params** utility to change the **SubscriberLifeCycle** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To enable BRM to use custom service life cycles:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsCustomer bus_params_customer.xml
   ```
3. In the file, change **disabled** to **enabled**:

   ```
   <SubscriberLifeCycle>enabled</SubscriberLifeCycle>
   ```

4. Save the file as **bus_params_customer.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_customer.xml
   ```

6. Stop and restart the CM.

Adding SLM Entries to the CM pin.conf File

To support custom service life cycles, BRM needs the following entries in the CM **pin.conf** file:

- **- cm_cache fm_bill_utils_business_profile_cache**
- **- cm_cache fm_bill_template_cache**
- **- cm_cache fm_cust_lifecycle_config_cache**
- **- cm_cache fm_cust_statemap_config_cache**

If these entries are not in your CM **pin.conf** file, you must add them.

To add the CM **pin.conf** entries for custom service life cycles:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
2. Add the following entries to the **pin.conf** file:

   ```
   - cm_cache fm_bill_utils_business_profile_cache number_of_entries, cache_size,
   hash_size
   - cm_cache fm_bill_template_cache number_of_entries, cache_size, hash_size
   - cm_cache fm_cust_lifecycle_config_cache number_of_entries, cache_size, hash_size
   - cm_cache fm_cust_statemap_config_cache number_of_entries, cache_size, hash_size
   ```

   where:

   - *number_of_entries* is the following:
     - For **fm_bill_utils_business_profile_cache**, the total number of business profiles (**/config/business_profile** objects), including service life cycle business profiles, you plan to create in your system. See "Associating Services with Custom Life Cycles" for more information.
     - For **fm_bill_template_cache**, the total number of service validation templates (**/config/template/service** objects), including those defined in service life cycle business profiles, you plan to create in your system. See "Associating Services with Custom Life Cycles" for more information.
     - For **fm_cust_lifecycle_config_cache**, the number of **/config/lifecycle_states** objects you plan to create in your system. Each object represents one custom service life cycle. See "Creating Custom Service Life Cycles" for more information.
     - For **fm_cust_statemap_config_cache**, the number of **/config/service_state_map** objects you plan to create in your system. The only valid value is **1**. See "Mapping States to Statuses" for more information.
   - *cache_size* is the sum of the sizes of the cached objects in bytes.
   - *hash_size* is the square root of *number_of_entries*.
3. Save and close the file.
4. Stop and restart the CM.

Mapping States to Statuses

Each service life cycle state must be mapped to a status (Active, Inactive, or Closed) in the default service life cycle. This mapping supports the following:

- Backward compatibility
- Custom service state changes triggered by account status changes
- Member service state changes triggered by subscription service status changes

> **Note:**
>
> You should not include both services that use the default service life cycle and services that use a custom service life cycle in a service group. If you do, the subscription service might be inactive while a member service is active.

A state can be mapped to only one status.

A status can be mapped to multiple states.

To create state-to-status mapping for the states in all your custom service life cycles, edit the service state mapping configuration file (*BRM_home*/**sys/data/config/config_service_state_map.xml**). See "About the Service State Mapping Configuration File" for more information.

After editing the configuration file, use the **load_config** utility to load the file's contents into the **/config/service_state_map** object in the BRM database. That object contains the state-to-status mapping for every custom service life cycle in your system. See "Mapping States to Statuses" for more information.

About the Default State of a Status

Consider the following situation:

- BRM needs a state for a service that uses a custom life cycle.
- Only the service status is known.
- The status is mapped to multiple states.

In this situation, BRM uses the status's default state, which is the state whose PIN_FLD_DEFAULT_FLAG field is set to **1** for that status in the **/config/service_state_map** object.

For example, suppose the Active (10100) status is mapped to the Active, Recharge Only, and Credit Expired states. If a state value is required for a service whose status (Active) is known but whose state is unknown, BRM uses the state in the PIN_FLD_STATES array of the **/config/service_state_map** object that contains the following values:

- PIN_FLD_DEFAULT_FLAG = **1**
- PIN_FLD_STATUS = **10100**

Note:

- When a status is mapped to multiple states, the default flag of only one of those states can be set to 1.
- Every service life cycle state defined in your system must have a transition to each status's default state. This enables BRM to complete a state transition when only the status to which the service must change is known. For example, if BRM receives a request to change a status from Active to Inactive and the current state of the service is Recharge Only, a transition from Recharge Only to Suspended (the default state for the Inactive status) must exist. If the transition is not defined, the transaction fails.

  State transitions are defined in the **config_lifecycle_states.xml** file. See "Creating Custom Service Life Cycles" for more information.

About the Service State Mapping Configuration File

You configure the state-to-status mapping for the states in all your custom service life cycles in the *BRM_home*/**sys/data/config/config_service_state_map.xml** file.

The mapping for each service life cycle state in your BRM system is configured in a **<States>** element, which has the following syntax:

```
<ObjectList>
   <ConfigObject>
      <DESCR>Description</DESCR>
      <NAME>Name</NAME>
      <STATES elem="0">
         <DEFAULT_FLAG>DefaultFlag</DEFAULT_FLAG>
         <LIFECYCLE_STATE>LifeCycleStateNumber</LIFECYCLE_STATE>
         <STATUS>StatusNumber</STATUS>
         <STATUS_FLAGS>StatusFlagNumber</STATUS_FLAGS>
      </STATES>
   </ConfigObject>
</ObjectList>
```

Table 8-2 lists the elements of the preceding syntax:

Table 8-2 Service State Mapping Elements

XML Element	Description	Notes
DESCR	(Optional) Character string that describes the state-to-status mapping.	Minimum length is 0 characters. Maximum length is 255 characters.
NAME	Character string used as the name of the mapping.	The name must be unique within your BRM system. Minimum length is 1 character. Maximum length is 255 characters.

Table 8-2 (Cont.) Service State Mapping Elements

XML Element	Description	Notes
STATES	Parent element of the child elements that map a state to a status. This element is mapped to the PIN_FLD_STATES array in the **/config/service_state_map** object. The array must contain all the service life cycle states in your system. It can contain states from multiple service life cycles. See "Mapping States to Statuses".	If you configure multiple **<STATES>** elements, the integer value of each element's **elem** attribute must be different. The values do not have to be sequential, and they have no relationship to the values of the **elem** attributes of other elements.
DEFAULT_FLAG	Integer that specifies whether this is the default state for the status to which it is mapped. A status can be mapped to multiple states, but only one of those states can be the status's default state. If a state is required when only a status value is available, BRM uses the default state for that status.	Specify one of the following values: • **1:** Default state. If multiple states are mapped to a status, only one of those states can have this value. • **0:** Not the default state.
LIFECYCLE_STATE	Integer that specifies the numeric ID of a custom service state.	Specify the unique ID of the custom service state to which this mapping applies. This ID is configured in the **config_lifecycle_states.xml** file. The predefined life cycle states are from 101 to 109, reserved life cycle states are from 110 to 115, and custom life cycle states are from 115 to 125. **Note:** Do not use predefined state numbers for custom life cycle states. Also, do not change the existing service state mapping and life cycle state mapping. For custom logic, use custom states and logic.
STATUS	Integer that specifies the numeric ID of the status in the default service life cycle to which this mapping applies.	Specify one of the following status IDs: • **10100:** Active • **10102:** Inactive • **10103:** Closed
STATUS_FLAGS	Integer that specifies the status flag to pass when a state change occurs. The status flag specifies the reason for the change.	When a service is reactivated, the value must match the value used in the previous state change. **Note:** STATUS_FLAGS values are listed in the *BRM_home***/include/pin_cust.h** file.

Mapping States to Statuses

> **Note:**
>
> A custom life cycle state should not be mapped to more than one status. Each status, however, can be mapped to multiple states.

To create state-to-status mapping:

1. Open the **config_service_state_map.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home*/**sys/data/config** directory.
2. Add a **<States>** element to the **<ConfigObject>** element.
3. In the **<ConfigObject>** element, specify values for the elements listed in Table 8-2.
4. Save and close the file.
5. Run the following command, which loads the changes into the **/config/service_state_map** object in the BRM database:

   ```
   load_config config_service_state_map.xml
   ```

   > **Note:**
   >
   > - The "load_config" utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.
   > - The **pin.conf** file must contain the following entry, which enables the utility to validate the XML file values:
   >
   >   ```
   >   - load_config validation_module libLoadValidSLM LoadValidSLM_init
   >   ```
   > - If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file. For example:
   >
   >   ```
   >   - load_config BRM_home/sys/data/config/config_service_state_map.xml
   >   ```

   The utility converts the XML file into one **/config/service_state_map** object.
6. Read the updated object with the **testnap** utility's **robj** command or with Object Browser to verify that all fields are correct.

   For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.
7. Stop and restart the CM.

Modifying State-to-Status Mapping

To modify state-to-status mapping:

1. Open the **config_service_state_map.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home*/**sys/data/config** directory.
2. Modify the **<States>** element in which the mapping is configured.

   See Table 8-2 for more information about the elements of state-to-status mapping.
3. Set the **configMode** attribute of the **<ObjectList>** element to one of the following values:
   - **replace:** (Default) Updates the existing **/config/service_state_map** objects.
   - **recreate:** Deletes the existing **/config/service_state_map** objects and creates new objects.
4. Save and close the file.
5. Run the following command, which loads the changes into the **/config/service_state_map** object in the BRM database:

```
load_config config_service_state_map.xml
```

> **Note:**
>
> - The "load_config" utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.
> - The **pin.conf** file must contain the following entry:
>
>   ```
>   - load_config validation_module libLoadValidSLM LoadValidSLM_init
>   ```
>
>   This entry enables the utility to validate the XML file values.
> - If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file. For example:
>
>   **load_config** *BRM_home*/**sys/data/config/config_service_state_map.xml**

The utility converts the XML file into one **/config/service_state_map** object.

6. Read the updated object with the **testnap** utility's **robj** command or with Object Browser to verify that all fields are correct.

   For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

7. Stop and restart the CM.

Deleting State-to-Status Mapping

To delete the state-to-status mapping from your BRM system, use the following **load_config** command:

```
load_config -r service_state_map
```

> **Caution:**
>
> Do not delete mapping that a service is using. Doing so will cause data corruption.

This command deletes the **/config/service_state_map** object from your BRM database.

See "load_config" for more information.

Associating Services with Custom Life Cycles

A service type (**/service** subclass object) can be associated with only one life cycle. By default, all service types use the default service life cycle.

To associate a service type with a custom service life cycle, add the following key-value pair to that service's validation template in the SLM business profile:

- **Key: lifecycle_obj**
- **Value:** Name of the custom service life cycle

The SLM business profile is configured in the **pin_slm_business_profile.xml** file.

> **Caution:**
>
> After a service type is in use in your BRM system, do not associate it with a different life cycle. If you do, state and status changes might fail and data might be corrupted.

See "Creating and Managing Business Profiles" for general information about business profiles.

To associate a service type with a custom life cycle:

1. Open the **pin_slm_business_profile.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home*/**sys/data/config** directory.

2. In the list of validation template IDs (**<TemplateID>** elements), add or delete the name and type of the validation template for the appropriate service.

   By default, this business profile contains these validation template IDs:

   ```
   <TemplateId name="ServiceTelcoGprs" type="/service/telco/gprs" />
   <TemplateId name="ServiceTelcoGsm" type="/service/telco/gsm" />
   <TemplateId name="ServiceTelcoGsmTel" type="/service/telco/gsm/telephony" />
   ```

   See "Defining Business Profiles" for more information about **<TemplateID>** elements.

3. In the list of key values (**<NameValue>** elements), add or delete key-value pairs to identify the bill units that belong to this business profile.

   See "Defining Business Profiles" for more information about **<NameValue>** elements.

4. In the list of validation templates (**<TemplateList>** parent element), add or delete the definition of the validation template (**<Template>** element) for the appropriate service.

5. Save and close the file.

6. Create a **/config/template/service** subclass for each service type in the list of validation template IDs (**<TemplateID>** elements) of the **slm_business_profile.xml** file.

   For example, to support the SLM business profile, create the following subclasses:

   - **/config/template/service/telco/gprs**
   - **/config/template/service/telco/gsm**
   - **/config/template/service/telco/gsm/telephony**

7. Run the following command, which loads the SLM business profile into a **/config/business_profile** object in the BRM database:

   ```
   load_pin_business_profile pin_slm_business_profile.xml
   ```

> **Note:**
>
> - The **load_pin_business_profile** utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.
> - If you do not run the utility from the directory in which the XML file is located, include the complete path to the file. For example:
>
>   ```
>   load_pin_business_profile BRM_home/sys/data/config/
>   pin_slm_business_profile.xml
>   ```
>
> See "load_pin_business_profile" for more information.

The **PIN_FLD_NAME** field in the **/config/business_profile** object containing the SLM business profile is set to **SLM**.

8. Read the updated object with the **testnap** utility's **robj** command or with Object Browser to verify that all fields are correct.

   For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

9. Stop and restart the CM.
10. (Optional) Make the SLM business profile your system's default business profile. See "Associating Bill Units with the SLM Business Profile".

Associating Bill Units with the SLM Business Profile

For an account to own a service that uses a custom life cycle, the account's bill unit must be associated with the business profile in which the service is linked to the custom life cycle.

By default, services are linked to custom life cycles in the SLM business profile. You can associate an account's bill unit with the SLM business profile in either of the following ways:

- Make the SLM business profile your system's default business profile. In this case, when an account is created, its bill unit is automatically associated with the default business profile. To enable this feature, run the **pin_bus_params** utility to change the **DefaultBusinessProfile** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.
- If the SLM business profile is not the system's default business profile, you can associate the SLM business profile with the bill unit after the account is created. See "Changing a Bill Unit's Business Profile" in *BRM Opcode Guide* for more information.

To make the SLM business profile your system's default business profile:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

3. In the file, change this element value to **SLM** (the name of the business profile configured in (**pin_slm_business_profile.xml**):

   ```
   <DefaultBusinessProfile>SLM</DefaultBusinessProfile>
   ```

4. Save the file as **bus_params_billing.xml**.
5. If the name of your service life cycle business profile is not SLM, do the following:

a. Open the **bus_params_billing.xsl** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home*/**xsd** directory.

b. Search the file for the following section:

```
<xsl:template match="bc:DefaultBusinessProfile">
```

c. In the following lines of that section, replace **SLM** with the name of your service life cycle business profile:

```
<xsl:when test="normalize-space(text()) = 'SLM'">
<xsl:text>SLM</xsl:text>
```

   For example, if the profile is named **XYZ**, the lines should look like this:

```
<xsl:when test="normalize-space(text()) = 'XYZ'">
<xsl:text>XYZ</xsl:text>
```

d. Close and save the file.

6. Load the XML file into the BRM database:

```
pin_bus_params bus_params_billing.xml
```

7. Stop and restart the CM.

About the Sample Prepaid Service Life Cycle

> **Note:**
>
> The sample prepaid service life cycle performs charging on specific charge configurations.

The default service life cycle configuration file (**config_lifecycle_states.xml**) contains a sample life cycle for prepaid services. The life cycle has the following states:

- **Preactive**: The starting state of the sample prepaid service life cycle. This state indicates that the subscriber has never used the service. Typically, the service is provisioned in this state.
- **Active**: A service automatically changes to this state when the subscriber first uses the service by making a call or sending an SMS. In this state, the service start date is set, and the subscriber can perform normal service activity.

  The length of time a service is active depends on its balance, usage, and plan type. If the balance reaches its credit limit or if the **Active** state expires, the service state automatically changes to **Recharge Only**.
- **Recharge Only**: An active service automatically changes to this state when its balance reaches its credit limit. In this state, the subscriber can receive calls and use free aspects of the service but cannot make prepaid calls. The subscriber can use top-ups to replenish the balance. If the subscriber does not replenish the balance before the state expires, the state changes to **Credit Expired**.
- **Credit Expired**: A service automatically changes to this state when its **Recharge Only** state expires. In this state, the subscriber cannot make or receive calls. If this state expires, the state changes to **Suspended**.

- **Dormant**: Enables service providers to identify unused subscriptions and to offer subscribers incentives to resume using their services. A CSR must set this state manually through Customer Center. A dormant service has the same available features as an active service. If a subscriber uses a dormant service, its state automatically changes to **Active**.
- **Fraud Investigated**: The service's account is being examined for evidence of fraudulent activity. A CSR must set this state manually through Customer Center. In this state, the subscriber cannot make or receive calls. This state can be changed to **Active** or **Closed**.
- **Suspended**: Implies that the subscriber intends to resume the service. In this state, no aspect of the service can be used. A subscriber might ask a CSR to suspend a service while she is on vacation or if she loses her mobile prepaid handset.
- **SuspendedActive**: Allows you to continue charging the subscriber during suspension. This state can be changed to **Active**.
- **Closed**: The final state of the sample prepaid service life cycle. In this state, no aspect of the service can be used.

About Triggering Sample Prepaid State Changes

Table 8-3 lists the triggers that initiate state changes in the sample prepaid service life cycle.

Table 8-3 Triggers for State Changes in the Sample Prepaid Service Life Cycle

Trigger	Change Performed By	From State	To State
Balance reaches credit limit	PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE	Active	Recharge Only
Balance replenished	PCM_OP_BAL_POL_CHECK_LIFECYCLE_STATE	Recharge Only or Credit Expired	Active
State expires	**pin_state_change** utility or CSR	Recharge Only	Credit Expired
State expires	**pin_state_change** utility or CSR	Credit Expired	Suspended
State expires	**pin_state_change** utility or CSR	SuspendedActive	Active
State expires	**pin_state_change** utility or CSR	Suspended	Closed
Fraudulent activity suspected	CSR	Active	Fraud Investigated
Fraudulent activity not found	CSR	Fraud Investigated	Active
Fraudulent activity found	CSR	Fraud Investigated	Closed
State expires	**pin_state_change** utility or CSR	Active	Recharge Only

For an overview of state-change triggers, see "Triggering State Changes in Custom Service Life Cycles".

About the Sample Business Rules

The sample prepaid service life cycle includes the following business rules:

- **REQ_ALLOWED**: Specifies whether a data usage request is allowed. If it is set to **Yes**, the other two sample business rules are evaluated before BRM determines whether to authorize or reject the request. If it is set to **No** or not set, the other two sample business rules are ignored and the request is rejected.
- **MO_ENABLED**: Specifies whether mobile-originated calls are allowed. Values are **Yes** and **No**. This rule can also apply custom rules that you configure.
- **MT_ENABLED**: Specifies whether mobile-terminated calls are allowed. Values are **Yes** and **No**. This rule can also apply custom rules that you configure.

When these rules are loaded into a **/config/lifecycle_states** object, they are combined into one rule named CALL_ALLOWED, whose values represent various combinations of the values of all three of the preceding rules. Table 8-4 shows the CALL_ALLOWED values for the sample prepaid service life cycle.

Table 8-4 CALL_ALLOWED Values for the Sample Prepaid Service Life Cycle

CALL_ALLOWED Value	MT_ENABLED	MO_ENABLED	REQ_ALLOWED
0	No	No	No
1	No	No	Yes
2	No	Yes	No
3	No	Yes	Yes
4	Yes	No	No
5	Yes	No	Yes
6	Yes	Yes	No
7	Yes	Yes	Yes

For an overview of business rules and information about creating custom business rules, see "Configuring Business Rules for Custom Service Life Cycles".

About the Sample Service State-to-Status Mapping

The default **config_service_state_map.xml** file contains the mappings shown in Table 8-5, which support the sample prepaid service life cycle.

Table 8-5 State-to-Status Mapping for the Sample Prepaid Service Life Cycle

<LIFECYCLE_STATE>	<DEFAULT_FLAG>	<STATUS>	<STATUS_FLAGS>
101 (Preactive)	0	10102 (Inactive)	0
102 (Active)	1	10100 (Active)	0
103 (Recharge Only)	0	10100 (Active)	0
104 (Credit Expired)	0	10100 (Active)	0
105 (Dormant)	0	10100 (Active)	0
106 (Fraud Investigated)	0	10100 (Active)	0
107 (Suspended)	1	10102 (Inactive)	0
108 (Closed)	1	10103 (Closed)	0
109 (SuspendedActive)	1	10100 (Active)	0

See "Mapping States to Statuses" for more information on service state-to-status mapping.

Part II
Configuring Account Currencies

This part provides information about managing account and system currencies in Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Managing System and Account Currencies
- Supporting EMU Currencies and the Euro

9

Managing System and Account Currencies

Learn how to set up system currencies and account currencies in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About System and Account Currencies

You specify two types of currency in your BRM system:

- **System currency**: The default currency for your entire database. It is typically the currency of the country where your BRM system is installed. Every BRM installation includes a system currency. The default system currency is US Dollars (ISO code 840).
- **Account currency**: The currency for a specific customer account. For example, a customer living in France has euro as the account currency. All payments, balance impacts, and billing adjustments use the customer's account currency.

  The account currency is set by the customer or customer service representative (CSR) when the account is created.

> **Note:**
>
> – You cannot change an account currency after the account has been created.
>
> – An account can have only one primary currency. If a customer requires two primary currencies, you need to create two accounts for that customer.
>
> – A nonpaying bill unit (**/billinfo** object) or account must use the same account currency as the parent account.

Your price list might include balance impacts in the system currency, which might be different from a customer's account currency. If so, BRM needs to convert the system currency that is used for rating to the customer's account currency before making an impact on the customer's account balance. Currency conversion takes effect in real time by using the following process:

1. When an event is rated, the balance impact is calculated by using the system currency.
2. The balance impact is converted to the customer's account currency and added to the balance.

By default, when rating an event, BRM tries to use a charge created for the account currency. If BRM cannot find a charge that uses the account currency, BRM searches for one that uses the system currency. If it cannot find one that uses the system currency, it continues to the next charge offer in priority order to rate the event.

Setting the System Currency

The default system currency is US Dollars (ISO code 840). To use US Dollars as the system currency, you do not have to do anything.

> **Note:**
>
> If your system currency is not US Dollars, you must set the system currency *during installation*. After installing BRM, you cannot change the system currency unless you use an SQL query, a method that BRM does not support.

To set the system currency to a currency other than US Dollars, do not preconfigure BRM during installation. Instead, edit the **$PIN_CONF_SYS_CURRENCY** entry in the **pin_setup.values** file and run the setup script.

Example of changing the system currency to the euro:

```
$PIN_CONF_SYS_CURRENCY = 978;
```

For more information, see "Installing BRM" in *BRM Installation Guide*.

Setting the Default Account Currency

The default account currency is used when a currency is not specified at account creation.

1. Open the Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**). *BRM_home* is the directory in which the BRM server software is installed.
2. Change the value of the **currency** entry.

   For example, to change currency from US Dollars to British Pounds, change 840 to 826:

   ```
   - fm_cust_pol currency 826
   ```
3. Save and close the file.
4. Stop and restart the CM.

Supporting a New Currency

To add a currency balance element, use PDC or Pricing Center. For more information, see "Configuring Balance Elements" in *PDC Creating Product Offerings* or Pricing Center Help.

To support a currency, you need to know its ISO currency code such as 840 for US Dollars. To find ISO currency codes, refer to the ISO 3166 standard.

> **Note:**
>
> The ISO currency codes are listed in the *BRM_home*/**include/pin_currency.h** file.

Using Multiple Currencies in Your Product Offerings

When you create your product offerings, you specify the currency for charges. Most charges use the system currency.

Customers can purchase charge offers that are rated with the system currency or with their account currency. The balance impacts, however, must use the account currency. There are two ways to accomplish this:

- Allow BRM to translate all currencies to the account currency.
- Create charge offers for each currency.

Converting currencies can result in fluctuating charges based on fluctuating exchange rates. Table 9-1 shows how exchange rates can cause variations in how much a customer pays for the same event, even if the amount charged by the charge offer stays the same.

Table 9-1 Effect of Exchange Rate Variation

Charge in System Currency	Exchange Rate	What Customer Pays in Account Currency
10	1 - 1.1	11
10	1 - 1.2	12
10	1 - .9	9

If you are contractually obligated to maintain consistent charges, you might need to create charge offers in multiple currencies.

For example, to charge for broadband access time, you could create charge offers that include broadband usage charge pricing with different currency balance impacts. There are advantages to this approach:

- You do not have to update the currency conversion rates.
- Your customers are billed in consistent amounts that do not change with exchange rates.
- When customers create an account, they see prices in a familiar currency.

> **Note:**
>
> To create multiple charge offers using different currencies, create a set of charge offers for one currency, and then use the Product Creation wizard to copy the charge offers for each currency.

Changing Currency Conversion Rates

BRM stores conversion rate information in the **/config/currency/conversionrates** object. You must update the object when conversion rates change.

The object can store conversion rates that apply to multiple time periods. For example, you could have different rates for the period from January 1st through July 31st, and the period from August 1st through October 31st.

At CM startup, BRM caches the **/config/currency/conversionrates** object. To support accurate conversion for suspended payments, the entire object, including all conversion rates, is cached. When you recycle a suspended payment, BRM uses the conversion rate that was valid for the period when the payment was originally made.

It's important therefore to configure the **/config/currency/conversionrates** object to include conversion rates for all time periods that might be relevant to suspended payments.

> **Note:**
>
> Include *only* conversion rates for time periods that are required to support suspended payments. If the **/config/currency/conversionrates** object includes a very large number of conversion rates, caching it can consume significant amounts of memory.

By default, the conversion rates for EMU currencies to euro and for euro to EMU currencies is preconfigured and loaded into the database when BRM is installed.

You define the conversion rates and their time periods by adding PIN_FLD_CUR_RATES_TIMEFRAMES arrays to the **/config/currency/conversionrates** object. There is a separate array for each time period. The rates themselves are defined in the PIN_FLD_CUR_CONV_RATES array in PIN_FLD_CUR_RATES_TIMEFRAMES. The time periods are defined by PIN_FLD_START_T and PIN_FLD_END_T fields.

> **Note:**
>
> Stop and restart the CM after editing the object.

10

Supporting EMU Currencies and the Euro

Learn how to manage Euro currencies in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Supporting EMU Currencies and the Euro

Countries that joined the Economic and Monetary Union (EMU) before February 2002, can only use the euro as their legal currency. Countries that joined the EMU after February 2002 can still use their national currency and the euro during the crossover period. By default, accounts in crossover countries use a primary account currency and a secondary account currency. These two account currencies handle the process of converting to the euro. They allow customers to pay in euros or in their native EMU currency until the conversion to the euro is complete.

When customers in EMU member countries create an account, they can select either the euro or the EMU as their primary account currency.

- When the primary account currency is an EMU currency, BRM automatically assigns the euro as the secondary account currency.
- When the euro is used as the primary account currency, the customer can choose an EMU currency as the secondary currency, or choose no secondary currency.

> **Note:**
>
> You can specify whether BRM requires a secondary currency when the euro is the primary currency. See "Changing Currency Conversion Rates".

Using the Euro and EMU Currencies in Your Price List

You cannot convert directly from one EMU currency to another EMU currency. You must use triangulation, that is, convert from one EMU currency to the euro and then from the euro to the other EMU currency. There are several implications of using triangulation:

- If you have customers from many countries, you should use the euro as the system currency. That way, all currency conversion is between the euro and EMU currencies, and never between two EMU currencies, because you cannot convert between account currencies.

If you must use an EMU currency as the system currency, you should make sure that customers who have an account currency different from the system currency cannot purchase charge offers by using the system currency. To do so, use the euro as the primary currency for those accounts.

- Restrict all charge pricing to use either the euro currency or the national EMU currencies, but not both. This ensures you do not redefine the EMU to euro conversion ratio and charge two different amounts for the same service.

Handling Euro Conversion Rounding Errors

Converting between the euro and EMU currencies can result in rounding discrepancies. To allow BRM to accept these rounding discrepancies without reporting an error, you define currency error-tolerance values for EMU currencies.

You set the currency error tolerance separately for each EMU currency. It can be based on a percentage or on minimum and maximum amounts of the total amount billed.

Use PDC or Pricing Center to set error tolerance.

Example of percentage tolerance

You might set the percentage error tolerance amounts for French Francs to 98%:

- If a customer pays in euros for a 100 FF charge, and the euro amount converts to 99 FF, the payment is accepted.
- If a customer pays in euros for a 100 FF charge, and the euro amount converts to 97 FF, the payment is not accepted.

Example of amount tolerance

You might set the minimum and maximum error tolerance amounts for French Francs to 3:

- If a customer pays in euros for a 50 FF charge, and the euro amount converts to 49 FF, the payment is accepted.
- If a customer pays in euros for a 50 FF charge, and the euro amount converts to 46 FF, the payment is not accepted.

Example of percentage and amount tolerance

If you set both an error tolerance percentage and a tolerance amount, the tolerance percentage overrides the tolerance amount. For example, you might set the percentage tolerance to 1% and the minimum amount tolerance to 5. The customer pays in euros for a 1000 FF charge, and the euro amount converts to 992. The percentage tolerance is met, so the payment is accepted, even though the amount minimum has not been met.

Rounding Errors for Overpayments and Underpayments

By default, BRM ignores conversion errors for overpayments and credits the excess to the customer's account. For underpayments, BRM uses the tolerance settings if the customer pays in the secondary currency. You can change how BRM uses the tolerance settings in PDC or Pricing Center.

In addition, you can specify how tolerance values are applied to primary and secondary currencies. See the **underdue_tolerance** and **overdue_tolerance** entries in the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).

Changing Supported Secondary Account Currencies

To limit the EMU currency choices, edit the **/config/currency/supportedcombinations** object.

Make the following changes to the object:

- Delete entries for currencies that you do not want displayed.
- Delete the **PIN_FLD_CREATED_T** field.
- Delete the **PIN_FLD_MOD_T** field.
- (Optional) To use the euro as the primary account currency without using a secondary currency, change the secondary currency required field to **0**:

```
2   PIN_FLD_CUR_SECONDARY_REQ   ENUM [0] 1
```

> **Note:**
>
> To set the default secondary currency, move the currency entry to the top of the list of supported currencies. If a secondary currency is required, and the secondary account currency is not supplied during account creation, the default secondary currency is the first supported currency listed.

Stop and restart the CM after editing the object.

To verify that you changed the fields, read the object by using the **testnap** utility or by displaying the **/config/currency/supportedcombinations** object in the Object Browser.

Part III

Sending Messages to Customers

This part describes how to send messages to your customers in Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Setting up Welcome Messages to Customers
- Sending Email to Customers Automatically
- Sending Messages to Customers through External Notification Applications

11

Setting up Welcome Messages to Customers

Learn how to send introductory messages to customers by email or on a Web page through Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- Sending a Welcome Email
- Setting Up a Welcome Web Page
- Using Variables to Include Customer Data in Welcome Emails and Web Pages
- Using Multiple Welcome Emails and Web Pages

Sending a Welcome Email

The default email message is located in *BRM_home***/sys/cm/welcome/default.welcome**. Use any text editor to modify the text. Before you can send the message you must:

- Configure the Email Data Manager (DM) that sends the message. See "Sending Email to Customers Automatically".
- Enable the email on the BRM server.

To enable the welcome message:

1. Open the Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**).
2. Change the value of the **new_account_welcome_msg entry** to **1**.
3. Save and close the file.

The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.

See also:

- Changing the Welcome Email Subject Line
- Changing the Welcome Email Sender Address
- Specifying the Welcome Email Location
- Disabling the Welcome Email

Changing the Welcome Email Subject Line

The subject line for the welcome email message is by default “Welcome to the Internet." To change this, edit the PCM_OP_CUST_POL_POST_COMMIT policy opcode.

Changing the Welcome Email Sender Address

The sender address is the email address that your customer sees as the sender address.

To change the welcome message sender address:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. (Optional) Change the **sender** entry. This entry changes the part of the email address that precedes the at sign (@); for example:

   sender@example.com

   The default is **postmaster**.
3. Change the **domain** entry. This entry changes the part of the email address that follows the at sign (@), for example:

   postmaster@*your_domain.com*
4. Save and close the file.

You do not need to restart the CM to enable this entry.

Specifying the Welcome Email Location

If you move the welcome message file from the default location, you must specify the new location.

To specify a new welcome message location:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Change the value of the **welcome_dir** entry.
3. Save and close the file.

The new value becomes effective immediately and applies to the next account created. The CM does not need to be restarted.

Disabling the Welcome Email

If the welcome message is enabled, but the Email DM is not running, you will see an error in the CM log file. If you do not run the Email Data Manager, disable the welcome message.

To disable the welcome message, do one of the following:

- Change the value of the **new_account_welcome_msg** entry in the CM configuration file (*BRM_home*/**sys/cm/pin.conf**) to **0** (default).
- Rename the default message file.
- Remove write permission on the default message file.

Setting Up a Welcome Web Page

The introductory Web page is an HTML file that you can display during Web-based account creation. The introductory message is not implemented by default. To display an introductory message, you must customize your automatic account creation method to call the PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode.

Use any text editor to edit the introductory message. The default introductory message is located in *BRM_home*/**sys/cm/intro/default.intro**.

If you move the introductory message file from the default location, you must specify the new location.

To specify a new introductory message location:

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Change the value of the **intro_dir** entry.
3. Save and close the file.

The new value becomes effective immediately and applies to the next account created. You do not need to restart the CM to enable this entry.

Using Variables to Include Customer Data in Welcome Emails and Web Pages

You can use variables to insert information based on the customer's input into an email or an HTML page. For example, to display the package that the customer selected, you use the **${price_plan}** variable in the introductory message:

```
You have selected the ${price_plan} package
```

If the customer selects the Basic package, the message says “You have selected the Basic package".

Table 11-1 shows the default variables.

Table 11-1 Variables in Welcome Messages

Use this variable	To insert this text	Example
${price_plan}	Package name	Basic
${plan_description}	Package description	Monthly broadband access
${promo_code}	Promotional code	Broadband1

You can define more variables by customizing the PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode.

Using Multiple Welcome Emails and Web Pages

You can create multiple welcome messages to display to specific groups of customers. For example, you might want different messages for each language.

The message that is displayed is based on a value entered during account creation.

> **Note:**
>
> This value is stored in the PIN_FLD_AAC_SOURCE field in the account object.

For example, you could identify which message to send by using the IP address that the user logged in to. In that case, an introductory message file name might be:

156.151.1.11.1700.intro

In this example:

- **156.151.1.11** is the IP address

- **1700** is the port number
- **intro** is the suffix

12

Sending Email to Customers Automatically

Learn how to use the Email Data Manager (DM) to send customer notifications and invoices in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

> **Note:**
>
> After the Email DM is running, you need to set up event notification to determine the events or times that trigger a customer email.

Configuring Email Data Manager

To configure Email Data Manager:

1. Open the CM configuration file (*BRM_home*/**sys/dm_email/pin.conf**).
2. Make sure these entries exist in the file:
   - **dm_pointer**
   - **fm_module**
   - **em_db**
3. If you changed the file, save it, and stop and restart the CM.

Configuring Email DM SMTP Server Details

By default, the Email DM sends email by using the *BRM_home*/**bin/brm_mail.py** script, which depends on the **DM_EMAIL_SMTP_SERVER** environment variable.

To configure the Email DM to send email using this script:

1. Set the **DM_EMAIL_SMTP_SERVER** environment variable to the host name and port of the SMTP server:

   ```
   setenv DM_EMAIL_SMTP_SERVER hostname:port
   ```

   where *hostname* is the host name of the SMTP server, and *port* is the port number on which the STARTTLS option is available.

   For example:

   ```
   setenv DM_EMAIL_SMTP_SERVER internal-mail-router.example.com:25
   ```

2. Stop and restart the Email DM.

Starting the Email Data Manager

To start the Email DM, run this command:

```
pin_ctl start dm_email
```

13

Sending Messages to Customers through External Notification Applications

Learn how to set up Oracle Communications Billing and Revenue Management (BRM) to send messages to your customers through an external notification application such as Oracle Communications Convergent Charging Controller.

Topics in this document:

> **Note:**
>
> To be able to send messages to your customers, BRM must be integrated with an external notification application through an Apache Kafka Server. For more information, see "About Integrating BRM with an Apache Kafka Server" in *BRM Developer's Guide*.

About Sending Messages to Customers through External Notification Applications

You can set up your system to send messages to your customers through an external notification application, such as Oracle Communications Convergent Charging Controller, when triggering events occur in BRM. For example, Convergent Charging Controller could send messages to your customers when they create an account, purchase a product, or cancel a subscription in BRM.

To do so, you configure BRM to send notifications containing details about the triggering event and the message to your external notification application. Your external notification application can use this information to send messages to your customers via email, text, or interactive voice response (IVR).

To set up BRM to send notifications to an external notification application, you configure the following:

- The types of events that trigger messages to be sent to your customers, such as product purchases.
- Which accounts can receive each type of message. For example, which accounts can receive account creation messages or which accounts can receive product purchase messages.
- When to send messages to your customers, such as immediately or at 15:00.
- How to send messages to your customers, such as through email or text messages.
- Whether to send messages several hours, days, weeks, or months *before* or *after* an event occurs in BRM. For example, you could send a reminder message to customers that their bill is due in four days or that their bill is one week past due.
- Whether customers are automatically opted in to receive a specific notification message type.
- Whether to enrich messages with your customers' preferences, such as their preferred language.

About the Types of BRM Events that Trigger Notifications

BRM sends notifications to your external notification applications when the following triggering events occur in BRM:

- Customer-created and BRM-initiated events that impact a customer's account immediately. For example, a customer creates an account, a customer purchases a product, BRM successfully processes a payment, or BRM issues a refund.
- Events that will impact a customer's account in the future, such as a bill becoming due. When configured to do so, BRM searches for the following types of events and generates a notification a specified amount of time *before* the event occurs:
  - A customer's balance is about to expire.
  - A customer's bill is almost due.
  - A collections action is about to be performed against a customer.
  - A customer's installment payment is almost due.
  - A customer's subscription is about to expire.
  - A customer's subscription is almost due for renewal.
  - A customer's service lifecycle state is about to change.
  - A custom event is almost due.
- Events that impacted a customer's account in the past and haven't been resolved, such as a bill being past due. When configured to do so, BRM searches for the following types of events and generates a notification a specified amount of time *after* the event occurs:
  - A customer's balance expired, and new funds still need to be added.
  - A customer's bill is past due.
  - A collections action was performed, but the customer has yet to respond.
  - A customer's installment payment is past due.
  - A customer's subscription has expired and hasn't been renewed.

- A customer's subscription is past the renewal date and has yet to be renewed.
- A customer's service lifecycle state was changed.
- A custom event has expired and still needs to be resolved.

You specify the types of BRM events that can trigger notifications to be sent to your customers by editing the *BRM_home***/sys/data/config/payloadconfig_kafka_sync.xml** and *BRM_home***/sys/data/config/pin_notify_kafka_sync** files. For more information, see "Configuring BRM to Publish Notifications to Kafka Servers" in *BRM Developer's Guide* for more information.

You specify which accounts can receive each notification type when you create notification specifications. See "Creating Notification Specifications".

About Message Delivery Times

You specify when to deliver messages to your customers by configuring the settings shown in Table 13-1.

Table 13-1 Delivery Time Configuration

Setting	Description
Preferred Time	The preferred time is unique to each account and is specified in 24-hour format, such as 08:00, 14:00, or 16:30. Your CSRs collect a customer's preferred time when they create or modify an account in Billing Care or a custom client application. A customer's preferred time is defined in an account's subscriber preferences. See "Defining Subscriber Preferences".
Scheduled Time	A scheduled time applies to a group of accounts that meet your criteria. BRM uses it only if an account does not have a preferred time. The scheduled time can be set to one of the following: • **Real Time**: Messages are sent immediately. • **Predefined Time**: Messages are sent at a specified time, such as at 15:00. • **Fixed Time In-Advance**: Messages are sent at a specified time, several minutes, hours, days, weeks, or months before a triggering event occurs. For example, at 12:00, one week before a bill is due. • **Fixed Time After Due Date**: Messages are sent at a specified time, several minutes, hours, days, weeks, or months after a triggering event occurs. For example, at 13:00, five days after an unpaid bill was due. **Note:** For **Fixed Time In-Advance** and **Fixed Time After Due Date**, you can specify only to enforce the systemwide silent period. This means that messages are sent immediately unless they occur during the silent period. For example, assume that the silent period is 23:00 – 08:00. If a triggering event occurs at 04:00, the message would be sent at 08:01 (after the silent period ends). If the triggering event occurs at 11:00, the message would be sent at 11:00. You define scheduled times by creating notification specifications in Billing Care or a custom client application. See "Creating Notification Specifications".
Allowable Delay	An allowable delay specifies that messages can still be delivered to your customers a few minutes or hours after their preferred time or scheduled time. For example, assume a customer's preferred time is 15:00, and the allowable delay is 45 minutes. In this case, messages can be sent to the customer between 15:00 and 15:45. You configure the allowable delay by using a business parameter. See "Setting Systemwide Values for Notifications".

Table 13-1 (Cont.) Delivery Time Configuration

Setting	Description
Silent Period	A systemwide silent period, such as 24:00 – 06:00, during which messages cannot be sent to your customers. Messages scheduled for the silent period are sent after the silent period ends. **Note:** Silent periods are used when a preferred time and scheduled time are not defined. They can also be enforced by the fixed time in advance and fixed time after due date options (when configured to do so). You configure the systemwide silent period by using a business parameter. See "Setting Systemwide Values for Notifications".
Silent Days	Silent days are dates, such as 1 January 2030, when messages cannot be sent to your customers. Messages scheduled for silent days are sent on the next available non-silent day. **Note:** Silent days are enforced for only the predefined time, fixed time in advance, and fixed time after due date options. They are ignored for the preferred time and real-time options. You configure the systemwide silent days by using a business parameter. See "Configuring Silent Days".

If you have configured when to deliver messages using all of these methods, BRM determines a message's delivery time using the settings in the following order:

1. Your customers' preferred time (plus any allowable delay).
2. The scheduled time (plus any allowable delay). The scheduled time enforces the silent period if configured to do so.
3. The systemwide silent period.

Table 13-2 shows when messages would be delivered to your customers based on the delivery times configured in the account's subscriber preferences, notification specifications, and business parameters. In this example:

- All fixed times in advance and fixed times after the due date enforce the systemwide silent period
- The silent period is from 23:00 through 09:00
- The allowable delay is one hour

> **Note:**
>
> The message delivery times shown are simplified. The actual delivery times would be later due to the processing time.

Table 13-2 Sample Delivery Times

Triggering Event Time	Subscriber's Preferred Time	Notification Specification Scheduled Time	Message Delivery Time	Description
08:30 Monday	08:00	Predefined Time: 10:00	08:30 Monday	Sent immediately, because the one-hour allowable delay for the subscriber's preferred time means that messages can be sent between 8:00 and 9:00.
09:00 Monday	07:00	Predefined Time: 10:00	07:00 Tuesday	Sent the next day at the subscriber's preferred time, because messages can only be sent between 07:00 and 08:00.
10:00 Monday	22:00	Predefined Time: 14:00	22:00 Monday	Sent at the subscriber's preferred time.
08:30 Monday	Not Defined	Predefined Time: 12:00	12:00 Monday	Sent at the predefined time, because the subscriber's preferred time is not set.
06:00 Monday	Not Defined	Real Time	06:00 Monday	Sent immediately after the event occurs. The silent period is ignored, because the real time option doesn't support silent periods.
23:59 Monday	Not Defined	Fixed Time In Advance: Silent Period Enabled	09:01 Tuesday	Sent after the silent period ends.
08:00 Monday	Not Defined	Fixed Time In Advance: 10:30	10:30 Monday	Sent at the fixed time in advance.
10:00 Monday	Not Defined	Fixed Time After Due Date: Silent Period Enabled	10:00 Monday	Sent immediately after the event occurs, because the triggering event occurred outside of the silent period.
08:10 Monday	Not Defined	Not Defined	09:01 Monday	Sent after the silent period ends.

About Message Delivery Methods

By default, BRM supports the following message delivery methods: email, SMS, and Interactive voice response (IVR). You can also create custom delivery methods. To do so, see "Creating Custom Delivery Methods".

> **Note:**
>
> If your company supports email delivery methods, you must configure Billing Care or your client application to collect your customers' email addresses. Likewise, if your company supports the SMS or IVR delivery methods, you must configure Billing Care or your client application to collect your customers' phone numbers.

You specify how to deliver messages to your customers by configuring the settings shown in Table 13-3. BRM determines the delivery methods for a customer by using the settings in the priority order shown.

Table 13-3 Delivery Method Configuration

Priority Order	Setting	Description
1	Preferred Delivery Method	The preferred delivery methods are unique to each account. Your CSRs collect customers' preferred delivery methods when they create or modify an account in Billing Care or a custom client application. A customer's preferred delivery methods are defined in an account's subscriber preferences. See "Defining Subscriber Preferences".
2	System Delivery Method	A system delivery method applies to a group of accounts that meet your criteria and is used only if an account does not have a preferred delivery method associated with it. You define the system delivery method when you create notification specifications in Billing Care or a custom client application. See "Creating Notification Specifications".

About Generating Notifications In Advance

You generate notifications in advance by running the **pin_gen_notifications** utility. The utility searches for objects that will expire during a specified time range, such as between 1 May 15:00:00 and 2 May 15:00:00, and then generates a notification event for each object meeting the criteria.

The utility determines the time range to search for by doing the following:

1. Retrieving a customer's *offset value*, which specifies how far in advance to generate notifications, from the **/config/notification_spec** object. For example, the offset value could be 5 hours, 3 days, or 1 week.
2. Calculating the starting expiration date and time to search for by adding the offset value to today's date and time:

   Starting expiration = Today's date and time + Offset value

   For example, if the utility runs on 10 January at 12:00:00 and the offset is 10 hours, the starting expiration would be 10 January at 22:00:00. If the offset is 7 days, the starting expiration would be 17 January at 12:00:00.
3. Determining the ending expiration date and time to search for by adding the time range associated with the offset unit to the starting expiration:

   Ending expiration = Starting expiration + Time range

   where Time range is:

   - 1 minute if the offset value is in minutes. For example: 12:00:00 through 12:01:00.
   - 1 hour if the offset value is in hours. For example: 12:00:00 through 13:00:00.
   - 24 hours if the offset value is in days, weeks, months, or years. For example: 10 January at 12:00:00 through 11 January at 12:00:00.

The utility then searches for all objects due or expiring between the starting and ending expiration. For example, if the utility runs on 10 January at 12:00:00 and the offset value is 6 hours, the utility would search for objects that expire between 10 January 18:00:00 and 10 January 19:00:00.

Table 13-4 shows examples of when a notification event would be generated if the **pin_gen_notifications** utility was run on 1 May at 15:00:00 with the specified offset values.

Table 13-4 Sample Search Dates and Times for In-Advance Times

Offset Value	Search Date
30 Minutes	1 May 15:30:00 – 1 May 15:31:00
5 Hours	1 May 20:00:00 – 1 May 21:00:00
3 Days	4 May 15:00:00 – 5 May 15:00:00
5 Days	6 May 15:00:00 – 7 May 15:00:00
1 Week	8 May 15:00:00 – 9 May 15:00:00
1 Month	1 June 15:00:00 – 2 June 15:00:00
10 Minutes (1)	1 May 15:10:00 – Current time

Note:

1. There is a special case for 10 minutes or less in advance. In this case, the time range is the current time plus the offset value. For example, for 8 minutes in advance, the time range would be the current time until 8 minutes later. Likewise, for 5 minutes in advance, the time range would be the current time until 5 minutes later.

Alternatively, you can use command-line options to specify an expiration time range when **pin_gen_notifications** searches for objects. See "pin_gen_notifications" for more information.

About Generating Notifications After an Event Occurs

You use the **pin_gen_notifications** utility to generate notifications after an unresolved event occurs, such as after an installment payment is 5 days past due. The utility determines the time range to search for by using the same process as for generating notifications in advance, except it calculates the starting expiration date and time by subtracting the offset value from today's date and time:

Starting expiration = Today's date and time – Offset value

For example, if the utility runs on 10 January at 14:00 and the offset value is 4 days, the utility would search for objects that expired between 6 January 14:00 and 7 January 14:00. If the offset value is 1 week, the utility would search for objects that expired between 3 January 12:00:00 and 4 January 12:00:00.

Table 13-5 shows examples of when a notification event would be generated if the **pin_gen_notifications** utility was run on 1 May at 15:00:00 with the specified offset values.

Table 13-5 Sample Search Dates and Times for After Events Occur

Offset Value	Search Date and Time
45 Minutes	1 May 14:15:00 – 1 May 14:16:00
8 Hours	1 May 07:00:00 – 1 May 08:00:00
5 Days	26 April 15:00:00 – 27 April 15:00:00
2 Weeks	17 April 15:00:00 – 18 April 15:00:00
1 Month	1 April 15:00:00 – 1 April 15:00:00
10 Minutes (1)	1 May 14:50:00 – 1 May 15:00:00

Note:

1. There is a special case for 10 minutes or less. In this case, the time range is when the utility was run minus the offset value. For example, for 8 minutes, the time range would be from when the utility was run until 8 minutes before. Likewise, for 5 minutes, the time range would be when the utility was run until 5 minutes before.

Alternatively, you can use command-line options to specify an expiration time range when **pin_gen_notifications** searches for objects. See "pin_gen_notifications" for more information.

About Events Occurring Outside of Delivery Times

Sometimes, triggering events occur when messages cannot be sent to your customers. For example, a customer's subscription is canceled at noon, but the customer's preferred delivery time is 10:00.

In these situations, BRM does not generate a notification for the triggering event. Instead, BRM creates a **/schedule/notification** object that specifies the type of notification to generate, the date and time to generate it, and the account to generate it for.

To generate notifications for these **/schedule/notification** objects, you must run the **pin_deferred_act** utility frequently. When the utility is run, it finds all **/schedule/notification** objects with a delivery time (plus any allowable delay) that matches the current time and then generates the specified notifications. For information about the utility, see "pin_deferred_act".

For example, John Baker creates an account on Monday at 14:00. Because his preferred delivery time is 09:00, BRM creates **/schedule/notification** object **12345** that specifies to generate an **/event/notification/account/create** notification on Tuesday at 09:00 for John Baker. When **pin_deferred_act** is run on Tuesday at 09:00, it finds **/schedule/notification** object **12345** and generates the **/event/notification/account/create** notification.

About Enriching Notifications with Additional Information

BRM adds the notification type and message body to all notifications. However, you can have BRM add more information to notifications before they are sent to your external notification application by enabling notification enrichment. See "Enabling Notification Enrichment".

When enrichment is enabled, BRM also adds the following information to notifications:

- The preferred language for messages, such as English
- The message delivery method, such as SMS, Email, or IVR
- The phone number to use for SMS and IVR messages

You can also have custom information added to notifications during the enrichment process. To do so, you must specify the fields to add to notifications using a business parameter. See "Adding Custom Fields during Enrichment".

> **Note:**
>
> If your system supports the Email delivery method, you must configure BRM to add a customer's email address during the enrichment process.

The following shows a sample outgoing notification for a customer whose balance of 200 MB is about to expire. The non-bold fields contain the message type and message body, and the bold fields are added to the notification during the enrichment process.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Notification version="1.0.0.0.0">
    <BalanceExpiryNotification>
        <NotificationType>BALANCE_EXPIRY_NOTIFICATION_EVENT</NotificationType>
        <PublicUserIdentity>0049100073</PublicUserIdentity>
        <Balances>
            <ResourceId>1000009</ResourceId>
            <ResourceSymbol>Mbu</ResourceSymbol>
            <ResourceName>MB Used</ResourceName>
            <ValidTo>1588291200</ValidTo>
            <CurrentBalanceQuantity>200</CurrentBalanceQuantity>
            <RecId>0</RecId>
        </Balances>
        <BalGrpObj>0.0.0.1 /balance_group 134546 0</BalGrpObj>
        <ServiceObj>0.0.0.1 /service 134546 0</ServiceObj>
        <AccountObj>0.0.0.1 /account 134546 0</AccountObj>
        <SubscriberPreferences>
            <SubscriberPreferencesInfo>
                <PreferenceName>Language</PreferenceName>
                <PreferenceValue>English</PreferenceValue>
            </SubscriberPreferencesInfo>
            <SubscriberPreferencesInfo>
                <PreferenceName>Channel</PreferenceName>
                <PreferenceValue>SMS,IVR,EMAIL:abcd@example.com</
PreferenceValue>
            </SubscriberPreferencesInfo>
            <SubscriberPreferencesInfo>
                <PreferenceName>Customer Level</PreferenceName>
                <PreferenceValue>Gold</PreferenceValue>
            </SubscriberPreferencesInfo>
        </SubscriberPreferences>
    </BalanceExpiryNotification>
</Notification>
```

In this example, the information added during the enrichment process specifies:

- **Language**: English
- **Channel**: SMS, IVR, and Email (to abcd@example.com)
- **PublishUserIdentity**: Phone number for SMS and IVR is 0049100073
- **Customer Level**: Gold

During enrichment, BRM populates the fields in the notification from an account's subscriber preferences, which are stored in **/profile/subscriber_preferences** objects. If no subscriber preferences exist for an account, BRM populates the fields from your notification specifications, which are stored in a **/config/notification_spec** object.

> **Note:**
>
> If a phone number is not provided in an account's subscriber preferences, BRM populates the **<PublicUserIdentity>** field using the information from a CM **pin.conf** entry. See "Enabling Notification Enrichment".

Configuring BRM to Send Notifications to External Notification Applications

To configure BRM to send notifications to your external notification applications:

1. Specify which products, subscriptions, and balances can trigger notifications in advance or after an event occurs when you define balance elements and products in Pricing Design Center (PDC):
   - Specifying Which Balance Elements Trigger Notifications
   - Specifying Which Expiring Products Trigger Notifications
   - Specifying Which Subscription Renewals Trigger Notifications

   > **Note:**
   >
   > BRM automatically generates notifications for bills that are almost due.
2. Add subscriber preferences to your customers' accounts. See "Defining Subscriber Preferences".
3. Create notification specifications. See "Creating Notification Specifications".
4. (Optional) Specify the days of the year *not* to send messages to customers. See "Configuring Silent Days".
5. (Optional) Specify systemwide settings, including the silent period, the acceptable delay time, and the notification calendar for silent days. See "Setting Systemwide Values for Notifications".
6. (Optional) Create any custom delivery methods for your customers. See "Creating Custom Delivery Methods".
7. (Optional) Enable notification enrichment. See "Enabling Notification Enrichment".
8. (Optional) Specify any custom information to add during the notification enrichment process. See "Adding Custom Fields during Enrichment".
9. Integrate BRM with an external notification application through a Kafka Server. See "Integrating BRM with External Notification Applications".
10. To generate notifications for events occurring in the future or the past, run the **pin_gen_notifications** utility hourly. See "Running the pin_gen_notifications Utility".
11. Run the **pin_deferred_act** utility frequently, such as hourly or every 15 minutes, to generate notifications for events that occurred outside of delivery times. See "pin_deferred_act".

Specifying Which Balance Elements Trigger Notifications

You can configure any currency or noncurrency balance element to trigger notification events when its validity period is about to expire or has already expired without new funds being added. For example, a notification event can be triggered three days before a customer's balance of free minutes expires or one week after a customer's $20 credit has expired.

To do so, specify your expiration preferences when you create a balance element or resource.

Configuring Balance Elements to Trigger Notification

You can configure a balance element to trigger notifications before the balance expires, after the balance expires, or for both.

- **Using PDC**. When creating a balance element in PDC, select **Notify Before Expiration**, **Notify After Expiration**, or both. For more information, see "Creating Balance Elements" in *PDC Online Help*.
- **Using the ImportExportPricing utility**. For each balance element defined in your import XML file, set the following elements:
  - **<expiryNotification>**: Set this to **true** to enable in-advance notifications or **false** to disable them. The default is **true**.
  - **<postExpiryNotification>**: Set this to **true** to enable post-expiration notifications or **false** to disable them. The default is **true**.

  For example, the following shows how to enable only in-advance notifications for Australian Dollar balances:

```
<balanceElements>
   <name>Australian Dollar</name>
   <description>Australian Dollar</description>
   <priceListName>Default</priceListName>
   <code>AUD</code>
   <numericCode>36</numericCode>
   <symbol>$</symbol>
   <expiryNotification>true</expiryNotification>
   <postExpiryNotification>false</postExpiryNotification>
   ...
</balanceElements>
```

  For more information, see "Importing and Exporting Pricing and Setup Components" in *PDC Creating Product Offerings*.
- **Using the load_pin_beid utility**. For each resource defined in your **pin_beid** file, set the **<ENABLE_EXPIRY_NOTIFICATION>** element to one of the following:
  - **0** to disable notifications for both in-advance and post-expiration notifications.
  - **1** to enable notifications for in-advance notifications only.
  - **2** to enable notifications for post-expiration notifications only.
  - **3** to enable notifications for both in-advance and post-expiration notifications. This is the default.

For example, the following shows how to enable only in-advance notifications for Canadian Dollar balances:

```
<BALANCES elem="124">
   <APPLY_MODE>1</APPLY_MODE>
   <BEID_STR_CODE>CAD</BEID_STR_CODE>
   <CONSUMPTION_RULE>0</CONSUMPTION_RULE>
   <NAME>Canadian Dollar</NAME>
   <STATUS>1</STATUS>
   <SYMBOL>Can$</SYMBOL>
   <ENABLE_EXPIRY_NOTIFICATION>1</ENABLE_EXPIRY_NOTIFICATION>
   ...
</BALANCES>
```

For more information, see "load_pin_beid" in *BRM Setting Up Pipeline Pricing*.

Specifying Which Expiring Products Trigger Notifications

You can configure a subscription to trigger notification events before or after it expires. For example, a notification event can be triggered one week before a customer's one-year mobile telephony subscription expires. To do so, specify your expiration preferences for the charge offers or products in the subscription.

Configuring Expiring Products to Trigger Notifications

You can configure a charge offer or product to trigger notifications before it is about to expire, after it has expired, or for both.

- **Using PDC**. When configuring a charge offer, select **Notify subscriber prior to offer expiration**, **Notify subscriber after offer expiration**, or both. See "Specifying Charge Offer Settings" in *PDC Online Help*.
- **Using the ImportExportPricing utility**. For each charge offer defined in your import XML file, set these elements:
  - **<expiryNotification>**: Set this to **true** to enable in-advance notifications or **false** to disable them. The default is **true**.
  - **<postExpiryNotification>**: Set this to **true** to enable post-expiration notifications or **false** to disable them. The default is **true**.

  For example, the following shows how to enable only in-advance notifications for a sample charge offer:

```
<chargeOffering xmlns:pdc="http://xmlns.oracle.com/communications/platform/
model/pricing">
    <name>Sample Charge Offer</name>
    <pricingProfileName>Product Offering</pricingProfileName>
    <priceListName>Default</priceListName>
    <obsolete>false</obsolete>
    <timeRange>0/inf</timeRange>
    <productSpecName>TelcoGsmTelephony</productSpecName>
    <offerType>SUBSCRIPTION</offerType>
    <priority>1</priority>
    <partial>false</partial>
    <expiryNotification>true</expiryNotification>
```

```
    <postExpiryNotification>false</postExpiryNotification>
</chargeOffering>
```

For more information, see "Importing and Exporting Pricing and Setup Components" in *PDC Creating Product Offerings*.

- **Using the loadpricelist utility**. For each product defined in your price list file, set the following elements:
  - **<expiry_notification>**: Set it to **yes** to enable in-advance notifications or **no** to disable them. The default is **yes**.
  - **<prod_expiry_notification_postdue>**: Set it to **yes** to enable post-expiration notifications or **no** to disable them. The default is **yes**.

  For example, the following shows how to enable only post-expiration notifications for a sample product:

```
<product type="subscription" partial="no"
date_range_impact_type="INSTANTIATED_DATE">
    <product_name>Sample Product</product_name>
    <product_code>code</product_code>
    <priority>1</priority>
    <permitted>/service/telco/gsm/telephony</permitted>
    <expiry_notification>no</expiry_notification>
    <prod_expiry_notification_postdue>yes</
prod_expiry_notification_postdue>
</product>
```

For more information, see "Using the XML Pricing Interface to Create a Price List" in *BRM Setting Up Pipeline Pricing*.

Specifying Which Subscription Renewals Trigger Notifications

You can configure a subscription to trigger notification events before or after it is due for renewal. For example, a notification event can be triggered one month before a customer's one-year mobile telephony subscription is due for renewal. To do so, specify your renewal preferences for the charge offers or products in the subscription.

Configuring Subscription Renewals to Trigger Notifications

You can configure a charge offer or product to trigger notifications before it is due for renewal, after it is due for renewal, or for both.

- **Using PDC**. When configuring a charge offer, select **Notify subscriber prior to offer renewal**, **Notify subscriber after offer renewal**, or both. See "Specifying Charge Offer Settings" in *PDC Online Help*.
- **Using the ImportExportPricing utility**. For each charge offer defined in your import XML file, set these elements:
  - **<subscriptionDueNotification>**: Set this to **true** to enable in-advance notifications or **false** to disable them. The default is **true**.
  - **<postSubscriptionDueNotification>**: Set this to **true** to enable post-expiration notifications or **false** to disable them. The default is **true**.

For example, the following shows how to enable only in advance-notifications for a sample charge offer:

```
<chargeOffering xmlns:pdc="http://xmlns.oracle.com/communications/platform/
model/pricing">
    <name>Sample Charge Offer</name>
    <pricingProfileName>Product Offering</pricingProfileName>
    <priceListName>Default</priceListName>
    <obsolete>false</obsolete>
    <timeRange>0/inf</timeRange>
    <productSpecName>TelcoGsmTelephony</productSpecName>
    <offerType>SUBSCRIPTION</offerType>
    <priority>1</priority>
    <partial>false</partial>
    <subscriptionDueNotification>true</subscriptionDueNotification>
    <postSubscriptionDueNotification>false</
postSubscriptionDueNotification>
</chargeOffering>
```

For more information, see "Importing and Exporting Pricing and Setup Components" in *PDC Creating Product Offerings*.

- **Using the loadpricelist utility**. For each product defined in your price list file, set the following elements:
  - **<subscription_due_notification>**: Set it to **yes** to enable in-advance notifications or **no** to disable them. The default is **yes**.
  - **<subscription_due_notification_postdue>**: Set it to **yes** to enable post-expiration notifications or **no** to disable them. The default is **yes**.

  For example, the following shows how to enable only post-expiration notifications for a sample product:

```
<product type="subscription" partial="no"
date_range_impact_type="INSTANTIATED_DATE">
    <product_name>Sample Product</product_name>
    <product_code>code</product_code>
    <priority>1</priority>
    <permitted>/service/telco/gsm/telephony</permitted>
    <subscription_due_notification>no</subscription_due_notification>
    <subscription_due_notification_postdue>yes</
subscription_due_notification_postdue>
</product>
```

For more information, see "Using the XML Pricing Interface to Create a Price List" in *BRM Setting Up Pipeline Pricing*.

Defining Subscriber Preferences

You define your customers' preferences for receiving messages by adding subscriber preferences to their accounts. For example, you could configure BRM to collect the following subscriber preferences:

- The preferred language for messages, such as English or Spanish
- The message delivery method, such as SMS, Email, or IVR

- The preferred time for delivering messages, such as 16:00
- The phone number for sending SMS and IVR messages
- The email address for sending email messages
- Whether to opt out of receiving one or more specific message types
- Other custom preferences added by your company

For example, a customer could specify to receive messages in French at 15:00 local time through the email address abcd@example.com.

You add subscriber preferences to your customers' accounts by using one of the following:

- A custom client application calling the subscriber profile opcodes. See "Managing and Customizing Profiles" in *BRM Opcode Guide*.
- A custom client application calling the **Profile** operations in the Billing Care REST API. See *REST API Reference for Billing Care*.
- Billing Care. See "Viewing and Adding Subscriber Preferences" in *Billing Care Online Help*.

> **Note:**
>
> Billing Care does not collect subscriber preferences by default. You must configure Billing Care to display the subscriber preferences screens. See "Configuring Billing Care to Collect Subscriber Preferences".

Configuring Billing Care to Collect Subscriber Preferences

To be able to collect subscriber preferences using Billing Care, you must configure Billing Care to display the Subscriber Preferences dialog box and the fields that you want included in the dialog box. To do so, configure the **config_subscriber_preferences_map.xml** file and then load its contents into the **/config/subscriber_preferences_map** object using the **load_config** utility.

The default **config_subscriber_preferences_map.xml** file includes sample preferences fields, but you can add, modify, or remove them. Keep the **NotificationsOptOutList** and **NotificationsOptInList** preference fields if you want your customers to be able to opt in or out of receiving some message types.

> **Caution:**
>
> If you enabled enrichment, which adds language and channel information to outgoing messages, your file must include entries for **Language** and **Channel**. Also include entries for **Phone** and **Email** if your system supports the IVR, SMS, and email channels.

To configure Billing Care to collect subscriber preferences:

1. Open the *BRM_home***/sys/data/config/config_subscriber_preferences_map.xml** file.
2. In **<SUBSCRIBER_PREFERENCES>**, define the fields that you want added to the Subscriber Preferences dialog box.

   Table 13-6 describes the elements in **<SUBSCRIBER_PREFERENCES>**.

Table 13-6 Subscriber Preferences Elements

Element	Description
<NAME>	The name of the preferences field to display in the Billing Care Subscriber Preferences dialog box.
<SUBSCRIBER_PREFERENCE_ID>	A user-created ID for this preferences field.
<STRING_ID>	Used for localization. Billing Care uses this information to display the localized preference name.
<STR_VERSION>	The string field version.
<DEFAULT>	The default value for the preference field.
<TYPE>	The data type for the preference field value. Possible values include: • **1**: STR (alphanumeric) • **2**: INT (integer) • **3**: ENUM (one of an ordered list of possible values. You must provide an array for this selection.) See the **<VALUES>** element in this table. • **4**: DECIMAL • **5**: TSTAMP (timestamp) • **6**: TIME (time in 24-hour format *HH*:*mm*) • **7**: MULTI SELECT ENUM (a fixed list that is displayed as a string with check boxes or combo boxes. The list is stored as a string in CSV format.) • **262**: TIME RANGE (time range in 24-hour format: *HH*:*mm*-*HH*:*mm*. For example: 16:00-20:00)
<VALUES>	An array list of possible values for the field, used only for **ENUM** types.

For example, the following entries define a new preference field named **Include Current Date** with the possible values of **True** or **False** (default).

```
<ConfigObject>
   <DESCR>Subscriber Prefs Map</DESCR>
   <NAME>Subscriber Prefs Map</NAME>
   <SUBSCRIBER_PREFERENCES elem="x">
      <NAME>Include Current Date</NAME>
      <SUBSCRIBER_PREFERENCE_ID>10</SUBSCRIBER_PREFERENCE_ID>
      <STRING_ID>10</STRING_ID>
      <STR_VERSION>1</STR_VERSION>
      <DEFAULT>909</DEFAULT>
      <TYPE>3</TYPE>
         <VALUES elem="0">
            <VALUE>True</VALUE>
            <STRING_ID>908</STRING_ID>
            <STR_VERSION>998</STR_VERSION>
         </VALUES>
         <VALUES elem="1">
            <VALUE>False</VALUE>
            <STRING_ID>909</STRING_ID>
            <STR_VERSION>999</STR_VERSION>
         </VALUES>
   </SUBSCRIBER_PREFERENCES>
</ConfigObject>
```

3. Save **config_subscriber_preferences_map.xml**.
4. Load the file into the database by running the **load_config** utility:

   ```
   load_config config_subscriber_preferences_map.xml
   ```

   For more information, see "load_config" in *BRM Developer's Guide*.
5. Stop and restart the Connection Manager (CM).

To verify that the updated preference configurations were loaded, you can display the **/config/subscriber_preferences_map** object by using the Object Browser, or use the **robj** command with the **testnap** utility.

For more information about the **/config/subscriber_preferences_map** object, see *Storable Class Reference*.

Creating Notification Specifications

You create notification specifications to define which customers can receive a particular message. Each notification specification defines the following:

- The business event associated with the specification, such as **CustCreate**, **ProductTermination**, or **BillDue**. This indicates whether the message is generated when an account is created, when a product is canceled, when a bill is almost due, or so on.
- The delivery window, such as within 24 hours after the event occurs, several days before the event occurs, or several days after the event occurs.
- The specification's validity period, such as from June through December.
- Whether customers are automatically opted in, by default, to receive this notification message type.
- The criteria an account must meet to receive the message, such as the customer type, zip code, or payment method.
- For in-advance and post-expiration notifications, whether to aggregate multiple events for the same notification type into one notification message for the customer.
- The delivery method, which can be one or more of the following: text, email, or IVR.
- The scheduled time for delivery, such as immediately, at 14:00, or any time outside of the silent period.

> **Note:**
>
> BRM uses the scheduled time and delivery method only when they are not defined in an account's subscriber preferences.

For example, you can specify that account creation messages are sent to Canadian customers via email at 15:00 local time. Likewise, you can specify that bill due messages are sent to all customers via text at noon, five days before the bill is due.

You create notification specifications by using one of the following:

- A custom client application calling the notification opcodes. See "Notification Opcode Workflows" in *BRM Opcode Guide*.

- A custom client application calling the **Notification** REST endpoints in the Billing Care REST API. See *REST API Reference for Billing Care*.
- Billing Care. See "Notifications" in *Billing Care Online Help*.

> **Note:**
>
> If you use Billing Care, you must configure it to display the business events that your company supports. See "Configuring Billing Care to Display Business Events".

After you create a notification specification, its information is stored in a **/config/notification_spec** object. BRM can use the information from this object when generating notification events and when enriching messages.

Configuring Billing Care to Display Business Events

You can configure which business events are displayed in the Billing Care Notification screens and can be associated with a notification specification. To do so, you configure an XML file and then load its contents into a **/config/business_events** object using the **load_config_business_event** utility.

For more information about the utility, see "load_config_business_event".

To configure Billing Care to display business events:

1. Create a **config_business_events.xml** file by running the following command:

   ```
   load_config_business_event -w config_business_events.xml
   ```

   The contents from the **/config/business_events** object are written into the XML file.
2. Open the XML file in a text editor.
3. Add an **<EVENTS>** element for each business event that you want displayed in Billing Care.

   Table 13-7 describes the elements to configure under the **<EVENTS>** element.

Table 13-7 Business Event Elements

Element	Description
<DESCR>	A brief description of the business event.

Table 13-7 (Cont.) Business Event Elements

Element	Description
<EVENT_NAME>	The business event name. This name is displayed in Billing Care. For in-advance message delivery times, only these business events are supported: • **BalanceExpiry** • **ProductExpiry** • **SuscriptionRenewalDue** • **BillDue** • **CollectionsActionDue** • **InstallmentDue** • **ServiceLifeStateChangeExpiry** • A custom business event For post-expiration message delivery times, only these business events are supported: • **PostBalanceExpiry** • **PostBillDue** • **PostProductExpiry** • **PostSubscriptionRenewalDue** • **PostCollectionsActionDue** • **PostInstallmentDue** • **PostServiceLifeStateChangeExpiry** • A custom business event
<EVENT_TYPE>	Billing Care reads this field to determine which options to display in its **Trigger Time** list for the business event. The valid values are: • **IN-ADVANCE**: The business event is eligible for only the Fixed Time In Advance trigger time. • **REGULAR**: The business event is eligible for only these trigger times: Predefined Time and Real Time. • **AFTER-DUE-DATE**: The event is eligible for only the Fixed Time After Due Date trigger time.
<FLAGS>	Whether this business event can be published to the Kafka Server: • **1**: Enables the publishing of this business event to the Kafka Server. • **0**: Prevents this business event from being published to the Kafka Server.
<NAME>	A user-friendly name for the business event.
<STATUS_FLAGS>	An integer that specifies the external status. This field is not currently supported.
<NOTIFY_OPCODE>	The number of the opcode to call for creating the notification event. To find an opcode's number, see the opcode header files in *BRM_home*/**include/ops**. **Note:** This field applies to in-advance and after-due-date trigger times only.

Table 13-7 (Cont.) Business Event Elements

Element	Description
<NOTIFY_SEARCH_LEVEL>	The search query level for displaying the aggregation mode option in the Billing Care GUI: • **1**: Self-only level • **2**: Account level • **3**: Balance group level • **4**: Service level • **5**: Bill unit level This field applies only if **<EVENT_TYPE>** is set to **IN-ADVANCE** or **AFTER-DUE-DATE**. **Note:** You configure whether to send aggregated notifications for a specific business event in the notification specification.

The following shows sample entries for displaying the **WelcomeMsg**, **BalanceExpiry**, and **PostInstallmentDue** business events in Billing Care:

```
<ConfigObject>
   <POID>0.0.0.1 /config/business_events -1 0</POID>
   <ACCOUNT_OBJ>0.0.0.1 /account 1 0</ACCOUNT_OBJ>
   <DESCR>Publishing state (Enable/Disable) of individual business events</
DESCR>
   <HOSTNAME>-</HOSTNAME>
   <NAME>Business Events</NAME>
   <PROGRAM_NAME>-</PROGRAM_NAME>
   <EVENTS elem="0">
      <DESCR>Welcome Mail</DESCR>
      <EVENT_NAME>WelcomeMsg</EVENT_NAME>
      <EVENT_TYPE>REGULAR</EVENT_TYPE>
      <FLAGS>1</FLAGS>
      <NAME>Welcome Mail</NAME>
      <STATUS_FLAGS>0</STATUS_FLAGS>
      <NOTIFY_OPCODE>1301</NOTIFY_OPCODE>
      <NOTIFY_SEARCH_LEVEL>2</NOTIFY_SEARCH_LEVEL>
   </EVENTS>
   <EVENTS elem="1">
      <DESCR>Balance Expiration</DESCR>
      <EVENT_NAME>BalanceExpiry</EVENT_NAME>
      <EVENT_TYPE>IN-ADVANCE</EVENT_TYPE>
      <FLAGS>1</FLAGS>
      <NAME>BalanceExpiry</NAME>
      <STATUS_FLAGS>0</STATUS_FLAGS>
      <NOTIFY_OPCODE>3787</NOTIFY_OPCODE>
      <NOTIFY_SEARCH_LEVEL>2</NOTIFY_SEARCH_LEVEL>
   </EVENTS>
   <EVENTS elem="2">
      <DESCR>Installment payment past due</DESCR>
      <EVENT_NAME>PostInstallmentDue</EVENT_NAME>
      <EVENT_TYPE>AFTER-DUE-DATE</EVENT_TYPE>
      <FLAGS>1</FLAGS>
      <NAME>Installment payment past due</NAME>
      <STATUS_FLAGS>0</STATUS_FLAGS>
      <NOTIFY_OPCODE>3626</NOTIFY_OPCODE>
      <NOTIFY_SEARCH_LEVEL>1</NOTIFY_SEARCH_LEVEL>
```

```
    </EVENTS>
</ConfigObject>
```

4. Save **config_business_events.xml**.
5. Load the XML file into the database by running this command:

   ```
   load_config_business_event -n config_business_events.xml
   ```

6. Stop and restart the Connection Manager (CM).

To verify that the configurations details were loaded, you can display the **/config/business_events** object by using the Object Browser, or by using the **robj** command with the **testnap** utility.

For more information about the **/config/business_events** object, see *Storable Class Reference*.

Configuring Silent Days

You can prevent messages from being sent to your customers on certain days of the year. For example, you could prevent messages from being sent on special holidays, such as Diwali or New Year's Day. To do so, you create a notification calendar that lists the dates, such as 1 June 2023 or 4 October 2023, which BRM considers as "silent days".

When a silent day occurs, messages are delayed until the next non-silent day. For example, if 1 January is a silent day, a message scheduled for 1 January at 14:00 would instead be sent on 2 January at 14:00.

> **Note:**
>
> Silent days are enforced for only predefined times, fixed times in advance, and fixed times after due date. They are ignored for preferred times and real times. See "About Message Delivery Times".

You configure silent days by editing the **config_silent_days_calendar.xml** file and then loading it into the **/config/calendar/silent_days** object using the **load_config** utility.

To configure silent days:

1. Open the *BRM_home***/sys/data/config/config_silent_days_calendar.xml** file.
2. Edit the file to include one or more calendars.

   Table 13-8 describes the elements in the file.

Table 13-8 Silent Calendar Day Elements

Element	Description
<DESCR>	A brief description of the calendar.
<NAME>	The name of the calendar, which must be unique. The default calendar name is **NotificationSilentDays**. You can create one or more calendars. For example, you could create a North American calendar, a European calendar, and so on.

Table 13-8 (Cont.) Silent Calendar Day Elements

Element	Description
<CALENDAR_DOM>	The day of the month. Valid values are 1 through 31.
<CALENDAR_MOY>	The number representing the month. Valid values are 1 through 12.
<CALENDAR_YEAR>	The calendar year, such as 2023. Enter **0** to indicate the current year and every year in the future. For example, a value of **0** could mean 2022, 2023, 2024, 2025, and so on.

For example, the following entries create a calendar named **NotificationSilentDays** with the following silent days: 10 August 2023 and every New Year's Day (1 January).

```
<ConfigObject>
   <DESCR>Default Notification Silent Days</DESCR>
   <NAME>NotificationSilentDays</NAME>
   <CALENDAR_DATE elem="1">
      <!-- Random Holiday -->
      <CALENDAR_DOM>10</CALENDAR_DOM>
      <CALENDAR_MOY>8</CALENDAR_MOY>
      <CALENDAR_YEAR>2023</CALENDAR_YEAR>
   </CALENDAR_DATE>
   <CALENDAR_DATE elem="2">
      <!-- New Year's Day -->
      <CALENDAR_DOM>1</CALENDAR_DOM>
      <CALENDAR_MOY>1</CALENDAR_MOY>
      <CALENDAR_YEAR>0</CALENDAR_YEAR>
   </CALENDAR_DATE>
</ConfigObject>
```

3. Save the **config_silent_days_calendar.xml** file.
4. Load the file into the database by running the **load_config** utility:

   ```
   load_config config_silent_days_calendar.xml
   ```

   For more information, see "load_config" in *BRM Developer's Guide*.
5. If you have created multiple notification calendars, specify which one BRM should use. See "Setting Systemwide Values for Notifications".

To verify that the calendar days were loaded, you can display the **/config/calendar/silent_days** object by using the Object Browser, or use the **robj** command with the **testnap** utility. See "Reading an Object and Fields" in *BRM Developer's Guide*.

For more information about the **/config/calendar/silent_days** object, see *Storable Class Reference*.

Setting Systemwide Values for Notifications

You can set systemwide values for all notifications generated by BRM. To do so, you use the **pin_bus_params** utility to change the value of the following **System** business parameters:

- **NotificationSilentPeriod**: Specifies the systemwide silent period during which messages cannot be sent to customers. The default is 21:00 through 07:00.

- **AcceptableDelayTime**: Specifies how long after the delivery time that messages can still be delivered to your customers. For example, if a customer's delivery time is 15:00 and the allowable delay is 45 minutes, messages can be delivered to the customer between 15:00 and 15:45. The default is 2 hours.
- **SilentDaysCalendarName**: Specifies the name of the notification calendar to be used for Silent Days. The default is **NotificationSilentDays**.

See "pin_bus_params" in *BRM Developer's Guide* for information about the utility's syntax and parameters.

To set systemwide values for notifications:

1. Go to the *BRM_home***/sys/data/config** directory.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```

3. Set the silent period during which messages cannot be sent to your customers:

   ```
   <NotificationSilentPeriod>silent</NotificationSilentPeriod>
   ```

   where *silent* is the time span in 24-hour time and *HH*:*mm*-*HH*:*mm* format. For example, to prevent messages from being sent from 11:00 p.m. to 8:00 a.m., set *silent* to 23:00-08:00.
4. Set how long after the delivery time that messages can be sent to your customers:

   ```
   <AcceptableDelayTime>delay</AcceptableDelayTime>
   ```

   where *delay* is the amount of time in *HH*:*mm* format. For example, enter **00:45** for 45 minutes, and enter **02:30** for 2 hours and 30 minutes.
5. Specify the notification calendar to use for determining the silent days:

   ```
   <SilentDaysCalendarName>calendar</SilentDaysCalendarName>
   ```

   where *calendar* is the name of the calendar.
6. Save the file as **bus_params_system.xml**.
7. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_system.xml
   ```

8. Stop and restart the Connection Manager (CM).

Creating Custom Delivery Methods

By default, BRM supports the following message delivery methods: email, SMS, and IVR. You can add custom delivery methods or modify the existing ones. To do so, edit the **config_notification_delivery_methods.xml** file and then load its contents into the **/config/delivery_methods** object using the **load_config** utility.

To create custom delivery methods:

1. Open the *BRM_home***/sys/data/config/config_notification_delivery_methods.xml** file.
2. Add or modify delivery methods by using the **<DELIVERY_METHODS>** element.

Table 13-9 describes the elements under **<DELIVERY_METHODS>**.

Table 13-9 Delivery Method Elements

Element	Description
<NAME>	The name of the delivery method.
<TYPE>	The type number, which must be unique.
<DELIVERY_IDENTIFIER_TAG>	The subscriber preference name where BRM can look up the identifier for the delivery method. For example, where BRM can look up a customer's email address for the email delivery method. The subscriber preference name must match a value in the **/profile/subscriber_preferences** object's PIN_FLD_SUBSCRIBER_PREFERENCES.PIN_FLD_NAME field.

For example, the following entries create a **Twitter** delivery method. BRM will look up a customer's Twitter handle under the **TwitterHandle** preference name of the **/profile/subscriber_preferences** object:

```
<DELIVERY_METHODS elem="0">
   <NAME>Twitter</NAME>
   <TYPE>1</TYPE>
   <DELIVERY_IDENTIFIER_TAG>TwitterHandle</DELIVERY_IDENTIFIER_TAG>
</DELIVERY_METHODS>
```

3. Save the **config_notification_delivery_methods.xml** file.
4. Load the file into the database by running the **load_config** utility:

   ```
   load_config config_notification_delivery_methods.xml
   ```

   For more information, see "load_config" in *BRM Developer's Guide*.

To verify that the delivery methods were loaded, you can display the **/config/delivery_methods** object by using the Object Browser, or use the **robj** command with the **testnap** utility. See "Reading an Object and Fields" in *BRM Developer's Guide*.

For more information about the **/config/delivery_methods** object, see *Storable Class Reference*.

Adding Rollover Details to Subscription Renewals

You can configure BRM to add rollover details to **/event/notification/subscription/renewal** notification events before they are sent to the Kafka DM. To do so, you use the **pin_bus_params** utility to change the value of the **ApplyRolloverBeforeCycleFees** business parameter.

See "pin_bus_params" in *BRM Developer's Guide* for information about the utility's syntax and parameters.

To add rollover details to subscription renewals:

1. Go to the *BRM_home***/sys/data/config** directory.

2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

3. Set the **<ApplyRolloverBeforeCycleFees>** element:

   ```
   <ApplyRolloverBeforeCycleFees>value</ApplyRolloverBeforeCycleFees>
   ```

   where *value* is one of the following:

   - **enabled**: Rollover details are added to **/event/notification/subscription/renewal** notification events.
   - **disabled**: Rollover details *are not* added. This is the default.

4. Save the file as **bus_params_subscription.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the Connection Manager (CM).

Enabling Notification Enrichment

You can configure BRM to enrich outgoing notifications by adding or editing the following parameters in your Connection Manager (CM) configuration file (*BRM_home***/sys/cm/pin.conf**):

- To enable or disable enrichment:

  ```
  - fm_publish enable_preferences_enrichment value
  ```

  where *value* is set to **1** to enable enrichment, or to **0** to disable enrichment. The default is **0** (disabled).

- To allow enrichment for specific publishers:

  ```
  - fm_publish prefs_enabled_publisher_list publisher
  ```

  where *publisher* is a comma-separated list of the publisher database numbers for which enrichment is enabled. For example, you would enter **0.0.9.6** for the Kafka DM. This parameter is effective only when **enable_preferences_enrichment** is set to **1**. By default, enrichment is enabled for all publishers.

- To set where BRM can find an account's phone number when it is not set in an account's subscriber preferences:

  ```
  - fm_publish prefs_phone_no_location location
  ```

  where *location* is one of the following values:

  – **-1**: Retrieves an account's phone number from the **/service** object's PIN_FLD_LOGIN field.

  – **0** through **4294901760**: Retrieves an account's phone number from the **/service** object's PIN_FLD_NAME field under the PIN_FLD_ALIAS_LIST array. The number

entered corresponds to the array index number. For example, **25** specifies to retrieve the account's phone number from ARRAY [25].

Restart the CM for the changes to take effect.

Adding Custom Fields during Enrichment

When notification enrichment is enabled, BRM enriches outgoing notifications with the following information:

- Preferred language
- Channel
- Phone number (for SMS and IVR messages only)

You can configure BRM to add custom information during the enrichment process, such as a customer's email address and streaming threshold. To do so, specify the custom fields to add to the notification by using the **NotificationSubscriberPreferences** business parameter.

> **Note:**
>
> If the email channel is supported, you must configure BRM to add a customer's email address during enrichment.

To specify the custom information to add to outgoing notifications:

1. Go to the *BRM_home***/sys/data/config** directory.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```

3. Set the list of fields to add to outgoing notifications:

   ```
   <NotificationSubscriberPreferences>preference</
   NotificationSubscriberPreferences>
   ```

   where *preference* is a comma-separated list of field names from the **/profile/subscriber_preferences** object. For example, to add a customer's phone number, email address, and streaming threshold, you could enter the following:

   ```
   <NotificationSubscriberPreferences>Phone,Email,Streaming Threshold</
   NotificationSubscriberPreferences>
   ```

   By default, this field is empty. If this field is missing or empty, BRM adds only the preferred language, channel, and phone number (for SMS and IVR) to outgoing notification messages.
4. Save the file as **bus_params_system.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_system.xml
   ```

6. Stop and restart the Connection Manager (CM).

Integrating BRM with External Notification Applications

To be able to send messages to your customers, you must integrate BRM with an external notification application through an Apache Kafka Server. During the integration process, ensure that you configure:

- The **pin_notify_kafka_sync** file to include the notifications that your company supports.
- The **payloadconfig_kafka_sync.xml** file to include only the business events that you want to send to your end customers through the external notification application.
- The **dm_kafka_config.xml** file to map each business event included in your notification specifications to a Kafka topic.

  If your external notification application is Oracle Communications Convergent Charging Controller, create Kafka topics with the **<TopicDefinition>** element's **format** set to **XML**, and **style** set to **OC3CNotification**. For example:

```
<TopicDefinition name="NotificationTopic" format="XML"
style="OC3CNotification"/>
   <PayloadName>BillDue</PayloadName>
</TopicDefinition>
```

For more information, see "About Integrating BRM with an Apache Kafka Server" in *BRM Developer's Guide*.

Running the pin_gen_notifications Utility

To generate notifications for objects expiring in the future, run the following command:

```
pin_gen_notifications parameter [-search_level self|account|bal_grp|service|
billinfo]
```

To generate notifications for objects that have already expired, run the following command:

```
pin_gen_notifications parameter -after_expiry [-search_level self|account|
bal_grp|service|billinfo]
```

where *parameter* is one of the following:

- **-balance_expiry**: Balance that will expire
- **-product_expiry**: Products that will expire
- **-subscription_renewal_due**: Subscriptions that are due for renewal
- **-bill_due**: Bills that are due
- **-installment due_date|end_date**: Installment payments that are due or about to end
- **-svc_lifestate_change**: Service lifecycle states that will change
- **-collections_action** [*action*]: Collections actions that are due

  If you run **-collections_action** without the *action* option, the utility generates notification events for all types of collections actions.

You can include the *action* option to generate notification events for just one type of collections action, such as closing a bill unit or collecting a payment.

> **Note:**
>
> Include the **-search_level** parameter to consolidate multiple events that expire at the same time into a single notification that is sent to your external notification application. The external notification application can then send information about multiple expiring events in one reminder message to your customers.

For example, if you run the following command on 10 July and the in-advance time is 3 days, the utility generates notifications for all subscriptions that are due for renewal on 13 July:

```
pin_gen_notifications -subscription_renewal_due
```

Alternatively, you can specify to search for objects that expire between dates and times that you specify. To do so, include the **-start** *timestamp* and **-end** *timestamp* parameters at the command line, replacing *timestamp* with the date and time in the format *mm***/***dd***/***yyyy***-***HH***:***mm***:***ss*, such as 12/23/2023-18:22:45. The start and end times are inclusive and can represent times in the future or in the past.

```
pin_gen_notifications -bill_due -start timestamp -end timestamp
```

You can also generate notifications for POIDs listed in a file that expire in a specified amount of time from today. To do so, include the **-file** *filename* parameter, replacing *filename* with the name and location of the file:

```
pin_gen_notifications -bill_due -file filename
```

For information about the content and syntax of *filename*, see "Creating a File of POIDs".

For more information about the utility's syntax and parameters, see "pin_gen_notifications".

Creating a File of POIDs

When running the **pin_gen_notifications** utility, you can specify to search through a list of object POIDs in a file for objects that are about to expire or have already expired. To do so, create a file in pin_flist_format that includes the following fields:

- PIN_FLD_POID set to the type of object to search for
- PIN_FLD_RESULTS array, with the following fields:
  - PIN_FLD_POID set to the POID of the **/bill**, **/balance_group**, **/service**, **/purchased_product**, **/collections_action**, **/installment_schedule**, or **/service** object to search for
  - PIN_FLD_ACCOUNT_OBJ set to the POID of the **/account** object to send the message to

The following sections show sample content for each type of notification.

- Sample Product Expiration File
- Sample Subscription Renewal File

Sample Product Expiration File

The following shows sample file content for product expiration notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /service/email -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 1
1       PIN_FLD_POID         POID [0] 0.0.0.1 /service/email 84139 9
1       PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 83499 0
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 1
1       PIN_FLD_POID         POID [0] 0.0.0.1 /service/email 107522 9
1       PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 106748 0
0 PIN_FLD_RESULTS           ARRAY [2] allocated 20, used 1
1       PIN_FLD_POID         POID [0] 0.0.0.1 /service/email 107522 9
1       PIN_FLD_ACCOUNT_OBJ  POID [0] 0.0.0.1 /account 106748 0
```

Sample Subscription Renewal File

The following shows sample file content for subscription renewal notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /service/ip -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /service/ip 84139 9
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 83499 0
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /service/ip 107522 9
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 106748 0
```

Sample Bill Due File

The following shows sample file content for bill due notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /bill -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /bill 84139 9
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 83499 0
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /bill 107522 9
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 106748 0
```

Sample Balance Expiration File

The following shows sample file content for balance expiration notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /balance_group -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 84139 9
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 83499 0
```

```
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 1
1     PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 107522 9
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 106748 0
```

Sample Collections Action File

The following shows sample file content for collections action notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /collections_action -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /collections_action/late_fee
135030 8
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130949 0
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /collections_action/late_fee
132750 8
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130459 0
0 PIN_FLD_RESULTS           ARRAY [2] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /collections_action/
finance_charge 131830 3
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130949 0
0 PIN_FLD_RESULTS           ARRAY [3] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /collections_action/
finance_charge 134542 3
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130459 0
```

Sample Installment File

The following shows sample file content for installment notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /installment_schedule -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /installment_schedule 135030 8
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130949 0
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /installment_schedule 132750 8
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130459 0
0 PIN_FLD_RESULTS           ARRAY [2] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /installment_schedule 131830 3
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130949 0
0 PIN_FLD_RESULTS           ARRAY [3] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /installment_schedule 134542 3
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 130459 0
```

Sample Service Lifecycle State File

The following shows sample file content for service lifecycle state change notifications:

```
0 PIN_FLD_POID              POID [0] 0.0.0.1 /service/ip -1 0
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /service/ip 3421 6
0 PIN_FLD_RESULTS           ARRAY [1] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /service/ip 5678 6
0 PIN_FLD_RESULTS           ARRAY [2] allocated 20, used 11
1     PIN_FLD_POID           POID [0] 0.0.0.1 /service/ip 8472 6
```

Adding Custom Business Events for In-Advance and Post-Expiration Notifications

You can configure BRM to send notifications to an external notification application when a custom object is about to expire or is past expiration. For example, you could configure it to send in-advance or post-expiration notifications based on the value of the PIN_FLD_DUE_T field in a custom storable class. To do so, perform the following procedure:

1. Add the following fields to your custom storable class (named **/custom_class_1** in this procedure):
   - PIN_FLD_LAST_NOTIFICATION_T (timestamp): Specifies the last time this object was processed for expiration notifications.
   - PIN_FLD_LAST_NOTIFICATION_OFFSET (string): Specifies the offset the last time this object was processed.

   See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide* for information about adding storable class fields.
2. Create a custom search opcode, such as CUSTOM_PIN_GEN_SEARCH_OPCODE, that does the following:
   - Retrieves the MTA application configuration flist as input.
   - Copies the entire input flist as the output flist.
   - Forms a custom search flist and replaces PIN_FLD_SEARCH_FLIST in the output flist with the custom search flist.
   - Retrieves the business event name from the PIN_FLD_APPLICATION_INFO.PIN_FLD_CMD_LINE_STR input flist field.

   See "Defining New Opcodes" in *BRM Developer's Guide* for more information.
3. Create a custom Notification opcode for generating the following custom notification events: **/event/notification/custom/custom_class_1/pre_expiry** and **/event/notification/custom/custom_class_1/post_expiry**.

   Customize the opcode to do the following:
   - Search for objects matching the date ranges in the PIN_FLD_VALUE_RANGES input flist array.
   - Check the PIN_FLD_DELIVERY_STATUS and PIN_FLD_WHEN_T input flist fields.

     If PIN_FLD_DELIVERY_STATUS is **true** and PIN_FLD_WHEN_T is not passed, call the PCM_OP_ACT_USAGE opcode with the PCM_OPFLG_CALC_ONLY flag set.
   - Call the PCM_OP_NOTIFICATION_GET_LAST_NOTIFY_TSTAMP opcode for each qualified candidate to retrieve the value of the PIN_FLD_LAST_NOTFICATION_T and PIN_FLD_LAST_NOTIFICATION_OFFSET fields.
   - Mark the last notification offset timestamp (PIN_FLD_LAST_NOTIFICATION_OFFSET) in each candidate object to prevent it from being picked up again.
   - Create an input and output flist similar to the ones for PCM_OP_BILL_NOTIFY_BILL_DUE and PCM_OP_BAL_NOTIFY_BALANCE_EXPIRY.

4. (Optional) Configure the Billing Care Notification screens to display your custom business event. To do so, configure the **config_business_events.xml** file and load it into the database using the **load_config_business_event** utility.

   See "Configuring Billing Care to Display Business Events".

5. Configure the **pin_gen_notifications** utility to call your custom search opcode during the MTA_INIT_SEARCH stage. To do so, use **testnap** or Developer Center to populate the PIN_FLD_OPCODE_MAP array in the **/config/mta** object.

   For example, to configure the utility to call the CUSTOM_PIN_GEN_SEARCH_OPCODE at the MTA_INIT_SEARCH stage, you would write the following to the **/config/mta** object:

   ```
   0 PIN_FLD_POID                 POID [0] 0.0.0.1 /config/mta 203607 0
   0 PIN_FLD_CONFIG_MTA          ARRAY [0] allocated 3, used 3
   1    PIN_FLD_NAME               STR [0] "pin_gen_notifications"
   1    PIN_FLD_OPCODE_MAP        ARRAY [0]
   2        PIN_FLD_FUNCTION       STR [0] "MTA_INIT_SEARCH"
   2        PIN_FLD_NAME           STR [0] "CUSTOM_PIN_GEN_SEARCH_OPCODE"
   ```

   See "Configuring the MTA Policy Opcodes" in *BRM Developer's Guide* for more information.

6. Configure the event notification system to send your custom notification events to the Payload Generator EM:

   a. Add the following lines to your *BRM_home***/sys/data/config/pin_notify_kafka_sync** file:

      ```
      1301 0 /event/notification/custom/custom_class_1/pre_expiry
      1301 0 /event/notification/custom/custom_class_1/post_expiry
      ```

   b. Load the **pin_notify_kafka_sync** file into the BRM database:

      ```
      load_pin_notify -v pin_notify_kafka_sync
      ```

   See "Configuring Event Notification for Kafka Servers" in *BRM Developer's Guide* for more information.

7. Configure the Payload Generator EM to build your custom business event (**Custom1BusinessEvent**) for the Kafka server. To do so, define the **Custom1BusinessEvent** business event in the *BRM_home***/sys/eai_js/payloadconfig_kafka_sync.xml** file.

   See "Defining Business Events" in *BRM Developer's Guide* for more information.

8. Create a notification specification for **Custom1BusinessEvent** using Billing Care or a custom client application.

   See "Creating Notification Specifications".

9. Run the **pin_gen_notifications** utility with the **-custom** parameter on an hourly basis:

   - For generating in-advance notifications, run this command:

     ```
     pin_gen_notifications -custom Custom1BusinessEvent -verbose
     ```

   - For generating post-expiration notifications, run this command:

     ```
     pin_gen_notifications -custom Custom1BusinessEvent -after_expiry -verbose
     ```

   See "pin_gen_notifications" for more information about the utility's syntax and parameters.

Part IV
Managing Customer Purchases

This part describes how to manage the products and contracts purchased by your customers using Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Managing Customer Contracts
- Managing Purchased Charge Offers
- Managing Purchased Discount Offers
- Managing Purchased Bundles
- Configuring Bundle and Package Transitions

14

Managing Customer Contracts

Learn how to manage your customers' contracts in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Contracts

> **Note:**
>
> You can create contracts for any revenue recognition scheme, and no feature needs to be enabled to use it.

A contract is a subscription with terms your customers agree to when purchasing it. The terms define the commitment period and options for canceling and renewing the subscription. For example, a contract might have a 1-year commitment period, a $100 early termination fee, and an automatic renewal after 12 months. A contract's terms apply to all offers, services, and required bundles in the contract. Any optional bundles the customer purchases are governed by their own independent terms.

After your customers have purchased a contract, they can:

- Modify the contract's auto-renewal options
- Purchase any optional bundles that are associated with their contract
- Cancel optional bundles in their contract, incurring any early termination fees for doing so
- Cancel their contract before the end of the commitment period, if allowed by the contract's terms, incurring any early termination fees for doing so

When their contracts reach the end of the commitment period, they are either canceled or automatically renewed when you run the **pin_contracts** utility. See "Canceling Expired Contracts" and "Processing Contract Auto-Renewals".

About Managing Customer Contracts

You manage your customers' contracts by using:

- Billing Care. See "Assets" in *Billing Care Online Help*.
- A custom client application calling the Contract FM standard opcodes. See "Managing Contracts" in *BRM Opcode Guide*.
- A custom client application calling the **Subscriptions** operations in the Billing Care REST API. See *REST API Reference for Billing Care*.

> **Note:**
>
> After contracts are created, BRM does not support the following operations:
>
> - Modifying existing contracts
> - Suspending contracts
> - Transitioning packages and bundles
> - Using complex discounts
>
> Only flat discounts that apply directly to the currency amount of cycle fees or purchase fees is supported. Discount grants on noncurrency resources and discounts based on resource values are not supported.
> - Using Account Migration Manager
> - Transferring services
> - Performing multischema operations

About Canceling a Customer's Contract Early

Your customers can cancel their contract before the end of the commitment period if the contract's terms allow it. When customers cancel their contracts early, BRM cancels all bundles, offers, and services in the contract and automatically charges your customers any early termination fees.

Your customers can also cancel any optional, add-on bundles in their contract before the end of the commitment period, if it is allowed by the bundle's terms, while still retaining their contract.

> **Note:**
>
> Because the contract's terms govern required bundles, customers cannot cancel a required bundle and still keep their contract. If they cancel a required bundle, the entire contract is canceled.

For example, assume a customer's Online Learning contract includes the following:

- A one-year commitment term with a $20 fee for canceling early
- A required bundle for digital textbooks
- An optional bundle for online tutoring with a one-year commitment term and no fee for canceling early

If the customer cancels the Online Learning contract or the digital textbook service early, they must pay the $20 fee, and BRM cancels the entire contract, including the online tutoring service.

If the customer cancels only the online tutoring service early, they are not charged a fee and keep their digital textbook service.

Processing Contract Auto-Renewals

You use the **pin_contracts** utility to process all contract auto-renewals in BRM. The utility searches the BRM database for all contracts that have auto-renewal enabled and have reached the end of their commitment period and then renews the contract to the new subscription term. See "About Subscription Terms" in *PDC Creating Product Offerings*.

When a contract's term is renewed, its start date is updated to the current contract's end date, and its end date is extended for the new commitment period specified in the terms. For example, when a six-month contract with a January 1 start date and a June 30 end date is renewed, the new 6-month contract has a start date of July 1 and an end date of December 31.

To process all auto-renewal contracts that expire today, run the following command:

```
pin_contracts -renew
```

For more information, see "pin_contracts".

You should run **pin_contracts** with the **-renew** parameter regularly to ensure that contract auto-renewals are processed in a timely manner.

Canceling Expired Contracts

You use the **pin_contracts** utility to cancel the following:

- Non-renewable contracts that are at the end of their commitment period
- Auto-renewal contracts that have already been renewed and are at the end of their second commitment period

When an expired contract is canceled, BRM cancels all bundles, offers, and services associated with the contract.

To cancel all non-renewable contracts that expire today, run the following command:

```
pin_contracts -expire
```

To cancel all contracts that have already been auto-renewed and are at the end of their second commitment period, run the following command:

```
pin_contracts -expire-renewed
```

For more information, see "pin_contracts".

You should run **pin_contracts** with the **-expire** and **-expire-renewed** parameters regularly to ensure that contracts are canceled in a timely manner.

Customizing Contract Management

BRM stores information about your customer's contracts, such as a contract's start and end date, in **/subscriber_contract** objects. For information, see Storable Class Reference.

You can extend the **/subscriber_contract** storable class to store additional fields. To do so, see "Extending Subscriber Contracts" in *BRM Opcode Guide*.

You can also customize how contract management does the following:

- Validates contracts by using the PCM_OP_CONTRACT_POL_VALID_CONTRACT policy opcode.
- Prepares a contract by using the PCM_OP_CONTRACT_POL_PREP_CONTRACT policy opcode.
- Creates contracts by using the PCM_OP_CONTRACT_POL_POST_CREATE_CONTRACT policy opcode.
- Calculates penalty charges during contract cancellation by using the PCM_OP_CONTRACT_POL_CANCEL_CONTRACT policy opcode.

For more information, see "Contract FM Policy Opcodes" in *BRM Opcode Guide*.

15

Managing Purchased Charge Offers

Learn how to manage your customers' purchased charge offers in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- Canceling Charge Offers
- Enabling Charge Offer Purchases from Closed or Inactive Accounts
- Changing Charge Offer Quantity

Canceling Charge Offers

You can cancel a charge offer immediately or backdate the cancellation to a date earlier than the current date. You can also cancel a charge offer in the future by modifying the charge offer end times.

See these topics for more options:

- Canceling Charge Offers Without Charging a Cancel Fee
- Including Event Adjustments in Charge Offer Cancellation Refunds
- Calculating Conditional Discount Offers When Canceling a Charge Offer
- Deleting Canceled Charge Offers

You can cancel individual charge offers or cancel all charge offers in a bundle at one time by canceling the bundle.

> **Note:**
>
> Deferred cancellation is available only for the entire quantity of the charge offer in the account.

If a charge offer's cycle forward fee allows proration, the customer is credited for any cycle forward fees that have been charged for but not used. BRM prorates based on the pricing in effect at the time the charge offer was purchased.

When you close an account, all of the charge offers it owns are canceled.

> **Note:**
>
> - In the case of deferred charge offer cancellation, the cancellation is applied when the **pin_bill_day** billing script is run.
> - If the charge offer is contained in a required bundle, you must either transition the *package* containing the required bundle or close the service to cancel the charge offer.

Canceling Charge Offers Without Charging a Cancel Fee

You can specify the amount of time after a purchase that a charge offer can be canceled without charging the customer. For example, you might want to specify a charge offer cancel tolerance of 30 minutes, which would allow the CSR to cancel an incorrectly assigned charge offer within 30 minutes without charging the customer.

To specify a cancellation tolerance:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the billing instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates an XML file named **bus_params_billing.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_billing.xml.out**, set **CancelTolerance** to the number of minutes applicable:

   ```
   <CancelTolerance>value</CancelTolerance>
   ```

   The default value is **15**.

   Any other value entered in **CancelTolerance** is in minutes.

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM configuration.
4. Save and exit the file.
5. Rename the **bus_params_billing.xml.out** file to **bus_params_billing.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

You do not need to restart the CM to update this entry.

Including Event Adjustments in Charge Offer Cancellation Refunds

By default, when a charge offer is canceled, BRM does not apply event adjustments when calculating the refund amount.

To configure BRM to include event adjustments in charge offer cancellation refunds, run the **pin_bus_params** utility to change the **EventAdjustmentsDuringCancellation** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To include event adjustments in charge offer cancellation refunds:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

3. In the file, change **0** to **1**:

   ```
   <EventAdjustmentsDuringCancellation>1</EventAdjustmentsDuringCancellation>
   ```

4. Save the file as **bus_params_subscription.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.

Calculating Conditional Discount Offers When Canceling a Charge Offer

When canceling a charge offer, discounts that are based on conditions cannot be reliably calculated. When a charge offer is canceled, the discount calculated during monthly charges is recalculated because the condition values applied during the monthly charge may be different from the values applied at the time of charge offer cancellation.

For example, suppose a customer receives a 5% discount when the total monthly charge is less than $100, but receives a 10% discount when the total monthly charge is greater than $100. At charge offer purchase time, the total monthly charge is just the purchase amount, which is likely to be less than $100, so a 5% discount is applied. If the charge offer is canceled in the same month or in a subsequent month, but the total monthly charge is greater than $100, a 10% discount is applied as a prorated charge (if the discount is set to be prorated). This occurs because the charge offer cancellation occurs before the end of the month.

> **Note:**
>
> Conditional discounts are rarely used for purchase and cancellation events.

To ensure that conditional discounts are reliably calculated when canceling a charge offer, do one of the following:

- When you define a discount, select options to prorate the balance impact for the amount to ensure there is no refund at the time of charge offer cancellation.
- If the discount calculated at charge offer cancellation is not correct and the customer calls to get a resolution, the CSR can manually calculate the correct discount and the prorated refund amount.

Deleting Canceled Charge Offers

You might want to delete canceled charge offers from your database (for example, to improve performance by reducing the amount of data stored in the database).

Oracle recommends that you not delete canceled charge offers and discount offers because other external systems might also use them.

> **Note:**
>
> - Do not delete canceled charge offers if you use delayed billing. By default, BRM does not delete **/purchased_product** objects (charge offer instances) when you cancel them. When BRM is set up to use delayed billing, BRM requires information about the canceled charge offers to charge for the delayed events that were generated before the charge offer was canceled. To charge for events for canceled charge offers, the canceled charge offer instances must persist as **/purchased_product** objects, which is the default behavior.
> - Do not delete canceled charge offers if you rerate events. Events that use deleted charge offers cannot be rerated.
> - You cannot delete a charge offer if provisioning tags are defined for that charge offer. Charge offers with provisioning tags are updated by the provisioning system and therefore must remain in the BRM database.

To automatically delete **/purchased_product** objects when a charge offer is canceled:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the subscription instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_subscription.xml.out**, set **KeepCancelledProductsOrDiscounts** to **disabled**:

   ```
   <KeepCancelledProductsOrDiscounts>disabled</KeepCancelledProductsOrDiscounts>
   ```

   The default value is **enabled**. It keeps the deleted charge offers in the **/purchased_product** objects.

> **Caution:**
>
> BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM configuration.

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.
8. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Enabling Charge Offer Purchases from Closed or Inactive Accounts

By default, closed and inactive accounts cannot purchase charge offers. You may, however, want to make such purchases possible.

For example, you may want CSRs to create accounts that are closed, then purchase charge offers for the account. This allows the customer to verify that the charge offer is correct before charging the account. When the account is made active, the customer is charged.

When you purchase a charge offer for inactive or closed accounts, the offer's status must be inactive. If the offer's status is active, rating modules rate and apply charges for the offer. If you do not want charges to be applied, change the offer's status to inactive at the time of purchase, or use PDC or Pricing Center to set the offer's status to inactive in the bundle. When the account is activated, change the offer's status to active for charges to be applied.

To enable charge offer purchases from closed or inactive accounts:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```
3. In the XML file, set the value for **<DealPurchaseForClosedAccount>** to **enabled**:

   ```
   <DealPurchaseForClosedAccount>enabled</DealPurchaseForClosedAccount>
   ```
4. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_billing.xml
   ```

5. Stop and restart the CM.

Changing Charge Offer Quantity

You can change the quantity of a purchased charge offer; for example, a charge offer that gives one hundred included minutes as a sign-up bonus.

If you increase the quantity of a charge offer that has a discount offer on purchase fees, that discount offer is applied to the entire purchase rather than to each unit of the charge offer that is added to the account. For example, if you increase the quantity of a charge offer in an account from one to five and the discount offer on purchase fees is $5, the customer receives a discount offer of $5, not $20.

16

Managing Purchased Discount Offers

Learn how to manage your customers' discount offers in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Setting Discount Offer Status

The status of a discount offer can be **Active**, **Inactive**, or **Canceled**. A discount offer is not valid when it is inactive. The discount offer does not apply to any user events generated while it is inactive.

When you change the status of a discount offer, you specify the new status and the reason for the status change.

A discount offer's status can change when you:

- **Purchase a discount offer**: You can set the status to **Active** or **Inactive** in the case of deferred purchase.
- **Purchase an inactive discount offer**: You can later reactivate it by setting its status to **Active**.
- **Cancel a discount offer**: You set the discount offer status to **Canceled**. See "Canceling Discount Offers".
- **Change the status of an account or service that owns a discount offer**: You change the status of the discount offer to the same status. When you close an account, you cancel any discount offers it owns.
- **Change the status of a discount**: You can set the discount offer status to active or inactive.

Setting Discount Offer Purchase, Cycle, and Usage Start and End Times

You specify a discount offer's validity period for the account that purchases the discount offer by setting the purchase start and end times. The cycle and usage start and end times specify when to start and stop charging cycle forward fees and rating usage events by using the discounted charge.

You can modify a discount offer's purchase, usage, and cycle start and end times at the following times:

- When purchasing a discount offer.
- When canceling a discount offer. By default, the discount offer's purchase, cycle, and usage end times are set to the cancellation time. You can change the date the discount offer expires by modifying the discount offer's end time.
- When changing the status of a discount offer.

> **Note:**
>
> Do not change a discount offer's purchase, cycle, or usage start time after the discount has been applied to the account; otherwise, the discount will be applied incorrectly.

BRM does not allow you to backdate the discount offer's purchase, usage, or cycle end date prior to the discount offer's purchase start date or prior to the G/L posting date.

The cycle and usage periods must start after the purchase period start time and end before the purchase period end time.

If you configure BRM for time stamp rounding, BRM rounds all start and end times to 00:00:00 hours.

Enabling Mutually Exclusive Discount Offers

You configure discount offer exclusions to prevent a discount offer from being used or purchased when another discount offer or package is owned. You configure discount offer exclusions in either PDC or Pricing Center, and in a BRM server configuration file.

By default, discount exclusions are disabled in BRM. You can enable them by running the **pin_bus_params** utility to change the **ValidateDiscountDependency** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

When you enable discount exclusions, you have the following options:

- You can enable both discount offer-to-discount offer exclusion and discount offer-to-package exclusion only simultaneously.
- With discount offer to-package exclusions, you can disable the exclusion when a discount offer is purchased, but allow it to be enforced at run time when a discount is applied to a charge.

To enable discount exclusions in BRM:

1. Go to *BRM_home*/**sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

3. Find the following line:

   ```
   <ValidateDiscountDependency>disabled</ValidateDiscountDependency>
   ```

4. Change **disabled** to any of the following:
   - **discToDiscExcl:** Enable discount offer-to-discount offer exclusions.
   - **discToPlanExcl:** Enable discount offer-to-package exclusions.
   - **enableBothExcl**: Enable both discount offer-to-discount offer and discount offer-to-package exclusions.

- **disableDiscToPlanExclAndNoPurTimeValidation:** Disable exclusions between packages and discount offers systemwide. When this flag is set, at purchase time no dependency validations are performed.
- **enableBothExclAndNoPurTimeValidation:** Enable both discount offer-to-discount offer and discount offer-to-package exclusions, and do not support dependency validations at purchase time.
- **returnOnFirstExcl:** Use the first mutually exclusive discount offer or discount offer/package if BRM finds any such conflicts.

> **Note:**
>
> BRM uses the XML in this file to overwrite the existing **billing** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, the changes affect the associated aspects of the BRM billing configuration.

5. Save the file as **bus_params_billing.xml**.
6. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_billing.xml
   ```

7. Stop and restart the CM.

Configuring Remaining Charge Cycle Discounting

Remaining charge cycle discounting is used in the following situation:

1. A discount is applied to a recurring fee.
2. A second discount is applied mid-cycle. In this case, the second discount might be applied to the original charge instead of to the remaining charge.
3. If remaining charge cycle discounting is enabled, the original discount is backed out and the two discounts are evaluated based on priority.

If a second discount is purchased mid-cycle, the original discount is backed out and the two discounts are evaluated based on priority.

This process has the following consequences and limitations:

- Although this process is run for all cycle fee discounts that are applied mid-cycle, it can change the outcome only for cycle fee discounts that are set to Remaining Charge, and are prorated.
- This process locks the account involved in the discount offer applied during the entire process or refunding and reapplying discounts. The lock is released at the end of the transaction.
- If the discount is shared, all members of the discount sharing group are locked, but only if the sharing start time is set to start when the discount sharing group is created or a member is added (the **PropagateDiscount** business parameter is **enabled**). Locks are released for all members at the end of the transaction.
- This process does not retroactively change balances. When a discount that grants a noncurrency balance is applied mid-cycle, calls rated based on the noncurrency balance can be incorrect.

For example, a cycle fee discount grants 100 minutes and the subscriber uses all 100 minutes in the first week. The discount is then canceled mid-month. The discount amount is reevaluated, resulting in an adjusted grant of 50 minutes. However, the minutes already used by the subscriber beyond the new grant amount are not recovered because the calls have already been rated. In this case, you can rerate the calls for the account.

- This process uses the discount exclusion rules that are in effect at the time of the evaluation. For example, if an exclusion rule changes on the 15th of the month and discounts are reevaluated on the 20th, the exclusion rule that took effect on the 15th is used when reevaluating discounts for the entire cycle.

By default, remaining charge discounting of cycle fees is disabled. You can enable this process by running the **pin_bus_params** utility to change the **SequentialCycleDiscounting** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To enable remaining charge discounting of cycle fees:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

3. In the file, change **disabled** to **enabled**:

   ```
   <SequentialCycleDiscounting>enabled</SequentialCycleDiscounting>
   ```

4. Save the file as **bus_params_billing.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_billing.xml
   ```

6. Stop and restart the CM.

Canceling Discount Offers

You typically cancel a discount offer when you cancel the bundle in which it is bundled or when you close the account or service that owns the discount offer. You can also cancel a discount offer immediately or backdate the cancellation to a date earlier than the current date. A discount offer gets canceled automatically when the discount offer's purchase end date has been reached.

When you cancel a discount offer, you also prorate and charge cycle forward fees that may have been discounted for the period during which the discount offer is no longer valid.

To cancel a discount offer, you must specify the quantity to cancel:

- If the quantity is the same as the quantity purchased by the account or service, BRM cancels the discount offer.
- If the quantity is less than the quantity purchased by the account or service, BRM does not cancel the discount offer; it subtracts that quantity from the internal count variable within the **/purchased_discount** object that represents the discount offer.

By default, when you cancel a discount offer, BRM retains the **/purchased_discount** object that describes the discount offer with a status of **Canceled**. You can specify whether to delete **/purchased_discount** objects or retain them with a status of **Canceled**.

Changing the status of a discount offer to *canceled* sets the purchase, usage, and cycle end times to the cancellation time.

When you cancel a discount offer for an account or service, you also cancel the discount offer for each discount sharing group that the account or service owns. For information about discount sharing groups, see "About Charge and Discount Sharing Groups".

See these related topics:

- Configuring Discount End Dates During Mid-Cycle Cancellations
- Rating Delayed Events for a Canceled Discount Offer
- Changing the Status of Discounts Canceled in Mid-Cycle
- Deleting Canceled Discount Offers

Configuring Discount End Dates During Mid-Cycle Cancellations

When a discount is canceled in the middle of an accounting cycle and the proration for the discount is set to **Full discount**, you can configure two ways to handle the discount:

- By default, BRM sets the discount end date to the end date of the accounting cycle.
- You can configure BRM to cancel the discount immediately. In this case, BRM sets the discount end date to the cancellation date.

To enable this feature, run the **pin_bus_params** utility to change the **CancelFullDiscountImmediate** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To immediately cancel discounts canceled in mid-cycle:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

3. In the file, change **disabled** to **enabled**:

   ```
   <CancelFullDiscountImmediate>enabled</CancelFullDiscountImmediate>
   ```

4. Save the file as **bus_params_subscription.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.

Rating Delayed Events for a Canceled Discount Offer

By default, when you cancel a discount offer instance, you set the **/purchased_discount** object's status to **Canceled**. When you set up BRM to use delayed billing, BRM requires information about the canceled discount offers. BRM uses this information to rate the delayed events that were generated before the discount offer was canceled. To rate events for canceled discount offers, the canceled discount offers must not be deleted.

BRM does not delete canceled discount offers by default. To ensure that canceled discount offer objects are not deleted, do the following:

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
2. If necessary, set the value of the **keep_cancelled_products_or_discounts** entry to **1**.
3. Save and close the file.
4. Restart the CM.

5. Edit the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT policy opcode. By default, this policy opcode keeps **/purchased_discount** objects when they are canceled. Verify that you set the opcode to cancel, not delete, the discount offer.

When you change the status of a discount offer to **Canceled**, you also set its purchase, usage, and cycle end dates to the cancellation date.

Changing the Status of Discounts Canceled in Mid-Cycle

When an entire discount is applied to discounts that are canceled in the middle of a cycle, BRM sets the discount to expire at the end of the cycle, but its status remains active.

To change the status of expired discounts from active to canceled, you must run the **pin_discount_cleanup** utility with the **-m** parameter.

You can run this utility daily or add it to the **pin_bill_day** utility to be run automatically.

Deleting Canceled Discount Offers

You might want to delete canceled discount offers from your database (for example, to improve performance by reducing the amount of data stored in the database).

> **Note:**
>
> - Do not delete canceled discount offers if you use delayed billing.
> - Do not delete canceled discount offers if you rerate events. Events that use deleted discount offers cannot be rerated.

To delete **/purchased_discount** objects when you cancel a discount offer:

1. Go to *BRM_home***/sys/data/config**.
2. Use the following command to create an editable XML file from the subscription instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_subscription.xml.out**,*set* **KeepCancelledProductsOrDiscounts** to :

   ```
   <KeepCancelledProductsOrDiscounts>disabled</KeepCancelledProductsOrDiscounts>
   ```

   The default value is **enabled**. It keeps the deleted charge offers in the **/purchased_discount** objects.

> **Caution:**
>
> BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM configuration.

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.
   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.
8. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

17

Managing Purchased Bundles

Learn how to manage your customers' purchased bundles in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Modifying Bundles

You use Billing Care or Customer Center to modify bundles owned by customer accounts (for example, to transition to a new bundle).

You can change the valid bundle period or the charge offers associated with the bundle, and you can define prerequisites, mutually exclusive relationships, or transitions for bundles.

Configuring Bundle Dependencies

You can set up bundle dependencies that determine which bundles you allow a customer to own.

To be able to check that no bundle violates the prerequisites and mutual exclusivity rules, you need to enable bundle dependencies validation.

In PDC or Pricing Center, you can define dependencies between bundles that set up the following relationships:

- **Prerequisites**: Specifies that an account must own a particular bundle to be able to purchase an additional bundle.

  A prerequisite can contain bundles of different services. For example, to own a GPRS bundle, an account must own a GSM bundle.

  A prerequisite bundle cannot contain item charge offers. When you create a package with alternate bundles, and if the base bundle contains an item charge offer, the purchase fails.
- **Required bundle**: Specifies whether bundles are optional or required for packages. *Required* bundles must be purchased when a package is purchased, whereas *optional* bundles can be added to an account at any time.
- **Mutual exclusivity**: Sets up a mutually exclusive relationship between two bundles, so if an account owns one bundle, it cannot own the other.
- **Allowed transitions**: Specifies which bundles or packages can serve as replacements for others.

  Transitions specify the bundles that customers can switch to and remain fully provisioned. While transitioning from one bundle to another, your customers retain their devices, such as phone numbers and services.

Customers owning bundles associated with a primary service may transition to other bundles associated with that primary service. The list of bundles displayed as available for transition are all associated with a specific primary service.

To enable the bundle dependencies validation.

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. *Uncomment* the **validate_deal_dependencies** entry to enable the bundle dependencies validation:

   ```
   - fm_utils validate_deal_dependencies 1
   ```

3. Save and close the file.

Bundle Dependency Validations Involving Inactive or Canceled Charge Offers or Discount Offers

By default, BRM does not perform bundle dependency validations for inactive or canceled charge offers or discount offers.

To enforce bundle dependency validations for inactive or canceled charge offers or discount offers, you need to configure BRM to do the following:

- Maintain any deleted charge offers in the **/purchased_product** objects to enable charge offer and discount offer validations.

  To do so, set the value of the **KeepCancelledProductsOrDiscounts** entry to **1** in the **business parameters** configuration file for Connection Manager. See "Deleting Canceled Charge Offers" for more information.

- Perform prerequisite and mutual exclusivity rule dependency validation.

  To do so, verify that the **validate_deal_dependencies** entry is uncommented in the **pin.conf** configuration file for Connection Manager.

- Perform bundle dependency validations on inactive or canceled charge offers and discount offers.

  To do so, enable the **ProductLevelValidation** business parameter in the **/config/business_params** object in the *BRM_home*/**sys/data/config/bus_params_subscription.xml** file.

  By default, **ProductLevelValidation** is set to **disabled**.

> **Note:**
>
> BRM performs the purchase time dependency validations between bundles irrespective of the value of the **ProductLevelValidation** parameter.

Enabling Bundle Dependency Validations for Inactive or Canceled Offers

When BRM is configured to enforce bundle dependency validations for inactive or canceled charge offers or discount offers and a bundle A is the prerequisite of bundle B:

- If bundle B is owned by the account, BRM will not allow cancellation or inactivation of any charge offers or discount offers of bundle A unless all the charge offers or discount offers of bundle B are in canceled or inactive state.

- If any charge offer or discount offer of bundle A is in the canceled state, bundle B cannot be purchased.
- If any charge offer or discount offer of bundle A is in the inactive state, bundle B can be purchased only in an inactive state; that is, all charge offers and discount offers of bundle B have to be purchased as inactive. All the charge offers or discount offers in prerequisite bundle A should be first activated before activating the charge offers or discount offers in bundle B.

> **Note:**
>
> BRM does not perform validations between the validity dates of the prerequisite and dependent bundles.

To enable this feature, run the **pin_bus_params** utility to change the **ProductLevelValidation** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To enable bundle dependency validations for inactive or canceled charge offers and discount offers:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

3. In the file, change **disabled** to **enabled**:

   ```
   <ProductLevelValidation>enabled</ProductLevelValidation>
   ```

4. Save the file as **bus_params_subscription.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

6. Stop and restart the CM.

Canceling Bundles

You can cancel all the charge offers in a bundle in one operation, or you can cancel individual charge offers. Canceling a bundle is particularly useful when a customer wants to upgrade. You can cancel all of the charge offers in the old bundle as a group before purchasing the new bundle for the customer's account. You can cancel a bundle immediately or backdate the cancellation of a bundle to a date earlier than the current date.

> **Note:**
>
> - After you cancel a bundle, you cannot reactivate the bundle or the charge offers it contains.
> - You cannot cancel a required bundle. Instead, either transition to a new package or close the service associated with the bundle.

18

Configuring Bundle and Package Transitions

Learn how to use transitions to determine the conditions that must be met to switch from one bundle or package to another in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Transitioning Bundles

You can set up transitions that determine which bundles can be purchased subsequent to another bundle.

To transition bundles, you use PDC or Pricing Center to set up transition rules, which are applied when you upgrade or downgrade a bundle for a customer. This enables you to limit the bundles that customers can switch to and still remain fully provisioned.

A bundle cannot be transitioned under the following circumstances:

- The new bundle is mutually exclusive to a bundle currently in the account.
- The account is missing a necessary prerequisite bundle.
- The bundle is associated with an inactive service.
- *The service being transitioned from* and the service being transitioned *to* are not the same service.
- The bundle is required in the associated package.

If a service was closed because it was associated with a package transition, you cannot change the closed status of the service.

To customize how to transition bundles, use the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL opcode. See *BRM Opcode Guide*.

Transitioning Packages

You specify the rules that govern the transition of an account from the source package to a target package. BRM imposes certain limitations on when accounts can transition to or from other packages. It requires you to define package transition rules for each package-to-package transition by manually configuring the transition rules in PDC or Pricing Center.

You can use the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN and PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN opcodes to customize package transitions. See *BRM Opcode Guide*.

When you transition accounts from one package to another package in BRM:

- If the transition type is a generation change, the two packages do not have to share the same primary service. If the transition type is an upgrade or a downgrade, the two packages must share the same primary service.
- If the source and target packages are a valid combination, you can configure the transition rule such that the source package retains the noncurrency grants as valid to the end of the cycle. To do so, set the value of PIN_FLD_FLAGS input to the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode to be PIN_SUBS_TRANSITION_CONTROL_ROLLOVER. (See **pin_subscription.h**.)
- BRM checks the status of any add-on bundles (that are not part of the source package) owned by the account. If all add-on bundles are canceled, BRM closes the associated services and transitions the account to the target package. If the add-on bundles are *not* canceled, BRM returns an error and retains the source package for the account. Therefore, you must first cancel all add-on bundles owned by the account *before* you transition the account to the target package.

The following restrictions apply to package transitions in BRM:

- You cannot backdate a package-transition or generation change to a prior period.
- If you delete a service from a package, BRM closes that service and sets its status to **PIN_STATUS_FLAG_DUE_TO_TRANSITION**. You cannot change the status of a closed service.
- BRM does not transfer extended rating attributes (ERAs) data during a package transition or a generation change. For example, if you have an account with ERA on friends and family and perform a package transition or generation change, the ERA data is not transferred to the new package.
- BRM, by default, retains the credit limits associated with the source package for an account when the source and target packages for that account are associated with the same balance group but each package has different credit limits. To set new credit limits for the account, you can customize PCM_OP_CUST_POL_TRANSITION_PLAN opcode by doing the following:
  1. Set the credit limit in the PIN_FLD_LIMITS array.
  2. Pass this array in the input flist to the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode.

About Defining Package Transition Rules

You can manually configure the package transition rule for each package-to-package transition in PDC or Pricing Center. For each such package transition rule you configure, BRM creates a **/transition** object to store the rules.

You can perform package transitions to any package without creating transition objects. This is a useful approach because for each package in the BRM system, you need to specify all other packages to which the package can transition. For example, if you have 300 packages and each package can transition to or from any other package, you define 89,401 (299 x 299) package transition rules, thus creating 89,401 **/transition** objects.

In addition, every time you add a package, you must define the transition rules for each existing package to point to the new package. As a result, the number of transition rules that you need to define in PDC or Pricing Center increases in proportion to any increase in the number of packages you support.

You can perform package transitions to any package without creating **/transition** objects to store the transition rules.

You use the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode to automatically enable package transitions to any package without **/transition** objects. See *BRM Opcode Guide*.

Defining a Generation Change for Packages

A *generation change* enables you to transition your customers between 2G (second generation) and 3G (third generation) wireless packages and services. Packages are called 2G or 3G depending on whether their primary service type runs on a 2G or 3G wireless network. A 3G wireless network is faster than a 2G network and can transmit video and two-way video telephone calls.

You can use any 2G or 3G service type, such as the following:

- **/service/telco/pdc**, a 2G service type based on the Personal Digital Cellular standard for digital mobile telephony.
- **/service/telco/imt**, a 3G service type based on the International Mobile Telecommunications (IMT) standard for 3G wireless communications.

Whether you are transitioning a customer from a 2G to a 3G package or from a 3G to a 2G package, both packages are valid all day on the day of transition. The package that is being phased out expires at 00:00:00 hours at the end of the transition day; the package being transitioned to becomes valid at 00:00:00 hours at the beginning of the transition day.

You set the transition rules between two packages. You can also set up standalone bundles as add-ons and specify that the transition rules apply to those bundles as well.

> **Note:**
>
> - When a transition between two packages is under way, no other transition is allowed between the two packages for the duration of the transition day.
> - You can backdate the subscription actions to a prior period in case of a generation change, but doing so can lead to incorrect results.

For the package replacing the current package, the following happens at 00:00:00 hours at the start of the transition day:

- The purchase, cycle, and usage start dates are set to 00:00:00 hours at the beginning of the transition day.
- All dependent services and bundles associated with the package are included in the transition.
- Any cycle fees for this package are charged from 00:00:00 hours at the beginning of the transition day to the end of the billing cycle.

> **Note:**
>
> Purchase and cancel fees can be configured to be waived during the transition.

The following happens at transition time or slightly after:

- All configured bundles are purchased for the service.
- The new service is provisioned.

For the package you are phasing out, the following happens at 00:00:00 hours at the end of the transition day:

- The current service is closed.
- Any dependent services are closed.
- All associated charge offers and dependent services are canceled.
- Forward and arrears cycle fees are prorated from the beginning of the billing cycle to the cancellation time.

Configuring Services for a Generation Change

By default, the service being phased out is active for one day, the day the generation change takes place. You can configure the number of days both services involved in a generation change are active.

To enable this feature, run the **pin_bus_params** utility to change the **ProdEndOffsetPlanTransition** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To modify the number of days the phased-out service is active:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the billing instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

   This command creates an XML file named **bus_params_billing.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In the **bus_params_billing.xml.out** file, change the value of the **ProdEndOffsetPlanTransition** to the number of days you want the phased-out service to remain active. The minimum number of days you can keep a phased-out service active is **1** and the maximum is **31**. The default is **10**. For example, if you want the service to remain active for 15 days:

   ```
   <ProdEndOffsetPlanTransition>15</ProdEndOffsetPlanTransition>
   ```

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing billing instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM subscription configurations.

4. Save and exit the file.
5. Rename the **bus_params_billing.xml.out** file to **bus_params_billing.xml**.

6. Run the following command, which loads the updated contents of the file into the BRM database:

   ```
   load_pin_business_profile -v bus_params_billing.xml
   ```

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

8. Stop and restart the Connection Manager (CM).

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Creating Custom Transition Types for Bundles and Packages

By default, BRM provides these transition types for packages and bundles:

- Upgrade
- Downgrade
- Generation Change

To add custom transition types for bundles and packages:

1. Open the *BRM_home***/sys/data/pricing/example/pin_transition_type** file in a text editor.
2. Add your custom transition types by using this syntax:

   ```
   TransitionIDNumber    TransitionString
   ```

   where:

   - *TransitionIDNumber* specifies the ID of the transition type. This value will be visible in the **/transition** object. Your custom ID numbers must be unique and greater than the number 100. However, they need not be in numerical order.
   - *TransitionString* specifies the transition name that is displayed in PDC or Pricing Center.

   For example, add the following line to create a custom transition type named RED:

   ```
   101      RED
   ```

3. Save and close the file.
4. Go to the directory in which you saved the **pin_transition_type** file and enter the following command:

   ```
   load_transition_type [TransitionTypeFile]
   ```

   where *TransitionTypeFile* specifies the name and location of the file that contains your custom transition types. By default, the utility uses the **pin_transition_type** file in the directory from which you run the utility.

> **Note:**
>
> For more information about the utility's syntax, see "load_transition_type".

5. Restart PDC or Pricing Center.

Your new transition types are loaded into the **/config/transition_type** object and are displayed in PDC or Pricing Center the next time you start it.

Part V

Managing Customer Deposits

This part provides information about how to collect and manage your customers' deposits in Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Managing Deposits

19

Managing Deposits

Learn how to manage your customers' deposits in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Deposits

Deposits are security amounts you can collect from customers to help avoid credit risks.

For example, you might want a customer deposit option for plans that include a modem or other equipment, or for international roaming plans.

About Configuring Deposits

Using Oracle Communications Billing Care, you create deposit specification profiles and deposit specifications that define rules and properties for deposits you create for customers. A specification includes a specification profile. An individual customer deposit includes a specification.

Each specification includes a single specification profile, but you can use the same profile with multiple specifications. You can use the same specification for any number of customer deposits.

Deposit specification profiles define the common business rules for the deposits in your organization, including:

- **Validity end date:** The date after which the deposit specification profile is no longer available for creating new deposit specifications. The profile doesn't expire for specifications already using it.
- **Release type:** The default for releasing a deposit to the customer after its validity expires. The options are prepayment and refund.
- **Refunds:** If refunds are allowed, if they need approval, and if there are associated fees.
- **Transfers:** Ability to transfer a deposit amount between the accounts.
- **Overrides:** Whether CSRs can override the configured charge at the time of purchase.
- **Exemptibility:** If CSRs can waive the deposit.
- **Interest:** If the deposit accrues interest, and the details of the interest amount and how it's calculated.

Deposit specifications define the charging and billing behavior for deposits, including:

- **Type:** Whether the specification is for accounts, devices, packages, or services.
- **Charge offer:** A charge offer with the purchase deposit event that determines the charge for deposits.
- **Validity end date:** The date after which the deposit specification is no longer available for creating deposits.
- **Billing:** Whether the deposit is billed immediately or with the next bill.
- **Credit limit:** Whether to enable increasing a customer's credit limit on the deposit amount.

When creating the deposit specification, you can save it as a draft. You can modify or delete draft specifications. A specification needs to be in active status before you can use it to create deposits, and you can't modify or delete active specifications. After a specification's end date is past, its status is expired.

About Creating and Managing Customer Deposits

Use Billing Care or a custom client application calling BRM deposit opcodes to work with customer deposits. For more information, see:

- "Deposits" in *Billing Care Online Help*
- "Deposit Opcode Workflows" in *BRM Opcode Guide*

To create deposits, you need a package or bundle containing a charge offer that has a purchase deposit event mapped to the charging rule of the deposit. Customers then purchase the package, bundle, or device with the deposit charge offer.

For example, a package or bundle includes a $50 per month connection fee, an iPhone, and a $500 security deposit. After the package is purchased, BRM charges the $50 connection fee and the $500 deposit.

If you use a custom client application instead of Billing Care, call the PCM_OP_DEPOSIT_PURCHASE_DEPOSIT opcode to purchase the deposit and charge the customer.

About Deposit Payments

Customers can pay for deposits in the following ways:

- Pay all or part of the deposit amount immediately.
- Pay the amount with the next billing cycle. In this case, the entire deposit amount is billed with other charges in the next bill on the customer’s next billing day. A customer pays the bill as part of their regular bill payment.
- Pay with an advance payment. If a customer already has a payment credit in the account from a previous overpayment, they can use that to pay for the deposit.

Customers can also use a combination of these methods. For example, they can pay a $50 deposit from a $30 advanced payment and a $20 credit card payment.

After purchasing the deposit, payments can be made for an already purchased but partially paid deposit, using standard BRM payment functionality.

About Deposit Reversals

If a deposit is purchased by mistake, you can reverse the deposit amount in Billing Care or by using the PCM_OP_DEPOSIT_REVERSE_DEPOSIT opcode.

After the deposit is reversed, the charge is negated with a credit of an equal amount. If any payment is made towards the deposit, the payments are available as credit. Customers can use the credit to pay other charges or you can refund it.

> **Note:**
>
> Ensure that the deposit is reversed as soon as possible after it is created or purchased. You can't reverse transactions after releasing or transferring a deposit, or after interest is calculated.

About Interest Calculations

BRM calculates deposit interest according to the rules set in deposit specification profiles. Deposits can use one of three types of interest calculations:

- **Simple:** The interest is calculated on the principal deposit amount.
- **Compound:** The interest is calculated on the principal deposit amount plus the already accumulated interest.
- **Absolute:** The interest is a specified amount instead of a percentage.

In deposit specification profiles, you set a percentage or amount, how often interest is calculated, whether to calculate interest on partial payments, and optionally a G/L ID for tracking interest payments.

You can calculate interest using the **pin_deposit_calc_interest** MTA application in the billing job. This utility calls the PCM_OP_DEPOSIT_ADD_INTEREST opcode to calculate interest.

> **Note:**
>
> When you release or transfer part or all of a deposit, interest is calculated on the released or transferred amount.

About Transferring Deposits

You can transfer a customer’s deposit amount from one service or account to another service or account. You can do this using Billing Care or the PCM_OP_TRANSFER_DEPOSIT opcode. When the transfer happens, BRM closes the deposit from which the amount is transferred and creates a new deposit.

For example, a customer purchases a plan or service. After a few months, you cancel the service and give the customer the option to either transfer the deposit amount to pay for another service under the same account or transfer the amount to another account, maybe to a friend or family member.

About Transferring Multiple Deposits Simultaneously

You can transfer multiple deposits simultaneously to other accounts or services. For example, if a service has been stopped, multiple customers subscribing to that service might want to transfer their deposit amount to another service.

This type of transfer is called a bulk transfer and runs as a batch job from the command line. The batch file for the bulk transfer defines the purchased deposits involved in the transfer. The file is in CSV format and is typically supplied by an external source program. Bulk transfers use the BRM multithreaded application (MTA) framework to process the batch file and perform the transfers defined in this file for each listed deposit. For information on the MTA framework, see "About the BRM MTA Framework" in *BRM Developer's Guide*.

When BRM processes a bulk transfer, it performs each transfer as a separate transaction. Therefore, there is no need to roll back the successful transfers if some transfers fail. BRM reports any deposit transfers that failed the bulk transfer in a log file so that you can examine them and correct the problem.

About CSV Files for Bulk Deposit Transfers

For a bulk transfer, you generate a CSV file that lists each deposit being transferred. BRM creates an input flist for each transfer in the CSV file.

Create a standard CSV file with commas as field delimiters, line breaks as record delimiters, and no blank lines. Use the following format for each record:

purchased_deposit_POID, target_account_POID, target_billinfo_POID, target_service_POID, transfer_amount

In this record:

- These fields are required: *purchased_deposit_POID*, *target_account_POID* or *target_service_POID,* and *transfer_amount*. Make sure to include a comma for each omitted field.
- If you omit *target_billinfo_POID*, BRM transfers the deposit to the account.

This is an example of records from a CSV file for bulk transfers:

```
0.0.0.1 /purchased_deposit 1702640 0, 0.0.0.1 /account 1720950 0, 0.0.0.1/billinfo
1724022 0, 0.0.0.1 /service 1727894 0, 1500

0.0.0.1 /purchased_deposit 1726823 0, 0.0.0.1 /account 17355482 0, 0.0.0.1 /billinfo
1732794 0, 0.0.0.1 /service 1724901 0, 10000
```

The first record in this example shows a transfer of 1500 to the target account.

Check the CSV file you create for format problems and spurious blank lines.

Running a Bulk Deposit Transfer

To do a bulk deposit transfer, run the **pin_deposit_transfer_deposit** utility with the CSV file you created.

> **Note:**
>
> To connect to the BRM database, **pin_deposit_transfer_deposit** needs a configuration file in the *BRM_home*/**apps/pin_deposits/pin_deposit_transfer_deposit** directory. Run **pin_deposit_transfer_deposit** from this directory.

1. Run the following command:

   ```
   pin_deposit_transfer_deposit -f input_file.csv
   ```

   If the CSV file is not in your working directory, use the full path to the file.

   ```
   pin_deposit_transfer_deposit -f /files/bulk_transfer/input_file.csv
   ```

2. Check the ${PIN_LOG_DIR}**/pin_deposits/pin_deposit_transfer_deposit.pinlog** file for any failed transfers.

   The bulk deposit transfer application generates an intermediate file in flist format. The default name of this file is **pin_mta_search.flist**.

About Releasing Deposits

If a service is canceled or the deposit is about to reach the end of its validity period, the subscriber can get back the deposit amount. You set this up in deposit specification profiles, using Billing Care or the PCM_OP_RELEASE_DEPOSIT opcode.

You can release deposits in one of the following ways:

- **Convert to prepayments:** All or part of the deposit pays other billed charges. Customers can use any extra amount to pay off future billed charges.
- **Refunds:** You refund the deposit to customers using standard payment methods. If the deposit specification is configured to require a refund review, then the refund request must be approved before you can issue it. You can use Billing Care or the PCM_OP_DEPOSIT_UPDATE_REFUND_REQUEST opcode to approve or reject a refund request.
- **Zeroize:** You can release the customer deposit if you need to settle a customer's account balance. For example, when a service is disconnected, the customer might want to close the account and settle the account balance. When this happens, the entire deposit amount is refunded to the subscriber after settling any billed and unbilled balances. The account balance then becomes zero.

  This method is not available in Billing Care.

About Refunding Deposits

BRM refunds deposits based on their configurations and release types.

Before refunding the deposit amount, validations are performed and, if required, an authorized person approves the refund request after reviewing it.

If any deposits have refund handling fees, the transaction is recorded to the G/L ID, a charge is applied on the deposit, the refund fee is deducted from the deposit amount, and the remaining amount is refunded to the customer.

About the Deposit Utilities

The deposit utilities perform the following actions:

- **pin_deposit_calc_interest**: Finds the purchased deposits for which interest must be calculated, then calculates the interest amount based on the specified interest type and frequency.
- **pin_deposit_transfer_deposit**: Transfers the deposit balance to a specified account or service.
- **pin_deposit_release_purchased_deposit**: Searches purchased deposits whose end date is less than the current system date and releases the deposit amount based on the release type.

See "Customer Management Utilities".

Updating Deposits

You can update the following in an existing deposit:

- **Validity:** For example, you can change the validity period from 6 months to 3 months by changing the end date.
- **Release type:** For example, you can change the release type from prepayment to refund.

You can update using Billing Care or the PCM_OP_DEPOSIT_UPDATE_DEPOSIT opcode.

You can also use PCM_OP_DEPOSIT_UPDATE_DEPOSIT to update the deposit amount. If the deposit amount needs to be increased or decreased, the system reverses the old deposit and purchases a new deposit with the new amount. Any payment collected for the old deposit is allocated to the new deposit. If the new deposit amount is larger and the previous payment insufficient, you collect an additional payment. See "About Deposit Payments".

You can't update the amount in Billing Care.

Any additional deposit amount that remains after allocation is refunded to the customer or allocated against any other billed charges through standard BRM business processes.

> **Note:**
>
> Ensure that the deposit is updated as soon as possible after it is created or purchased. No transactions can be reversed after a deposit is released or transferred or if the interest is calculated. This can also cause bad debt to the service providers.

Setting Up Your System to Support Deposits

To set up BRM to support deposits, do the following:

1. Create deposit specification profiles, deposit specifications, and deposits in Billing Care or your third-party tool. See "Deposits" in *Billing Care Online Help*.
2. Create nightly cron jobs to run the following utilities:
   - **pin_deposit_release_purchased_deposit**. See "About Releasing Deposits".
   - **pin_deposit_calc_interest**. See "About Interest Calculations".

- **pin_deposit_transfer_deposit**. See "About Transferring Deposits".

3. Customize the deposit process as needed by using the deposit policy opcodes. See "Customizing Deposit Management".

Customizing Deposit Management

You can customize deposit management with policy opcodes in these ways:

- Validate and implement deposit specification profile information by using the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION_PROFILE policy opcode.
- Validate and implement deposit specification information by using the PCM_OP_DEPOSIT_POL_VALID_SPECIFICATION policy opcode.
- Validate and implement any customizations while purchasing a deposit by using the PCM_OP_DEPOSIT_POL_VALID_PURCHASE_DEPOSIT policy opcode.
- Validate and implement any customizations while updating a deposit by using the PCM_OP_DEPOSIT_POL_VALID_UPDATE_DEPOSIT policy opcode.
- Validate and implement any customizations while collecting the deposit payments by using the PCM_OP_DEPOSIT_POL_VALID_COLLECT_PAYMENT policy opcode.
- Validate and implement any customizations while reversing the deposit by using the PCM_OP_DEPOSIT_POL_VALID_REVERSE_DEPOSIT policy opcode.
- Validate and implement any customizations while releasing the deposit by using the PCM_OP_DEPOSIT_POL_VALID_RELEASE_DEPOSIT policy opcode.
- Validate and implement any customizations while transferring the deposit by using the PCM_OP_DEPOSIT_POL_VALID_TRANSFER_DEPOSIT policy opcode.
- Modify the effective date for calculating interest for the purchased deposit by using the PCM_OP_DEPOSIT_POL_PRE_ADD_INTEREST policy opcode. Modify the calculated interest amount and interest period as needed with the PCM_OP_DEPOSIT_POL_POST_ADD_INTEREST policy opcode.
- Validate and implement any customizations while requesting a refund of the deposit amount by using the PCM_OP_DEPOSIT_POL_VALID_REFUND_REQUEST policy opcode.

For more information, see "Deposit Opcode Workflows" in *BRM Opcode Guide*.

Specifying the Minimum Deposit Amount to Refund

You can specify the minimum amount to provide as a refund. The deposit refund action validates the configuration if an amount is greater than the amount you specified.

The minimum value is expressed in terms of the account currency. By default, the minimum amount is 0.

1. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**) in a text editor.
2. Change the value of the **MinimumRefundValue** entry. For example, to process refund items only for amounts greater than 0:

```
- fm_deposit_pol MinimumRefundValue 10.00
```

3. Save the file.

You do not need to restart the CM to enable this entry.

Specifying the Minimum Days to Refund After Account Closure

You can specify the minimum number of days to refund a deposit after closing an account or service. The deposit refund action validates the configuration days. The refund request will be raised only if the current date is less than the number of days of the account or service termination date.

The minimum value is expressed in terms of days. By default, the minimum number of days is 0.

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**) in a text editor.
2. Change the value of the **RefundMechanismStartDays** entry. For example, to process the refund items only 3 days after the service closure date:

   ```
   - fm_deposit_pol RefundMechanismStartDays 3
   ```

3. Save the file.

You do not need to restart the CM to enable this entry.

Part VI

Managing Customer Groups

This part provides information about grouping customer accounts in Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Managing Account and Bill Unit Hierarchies
- Managing Charge and Discount Sharing Groups
- Managing Product Sharing Groups and Discount Offer Sharing Groups
- Managing Balance Monitoring Groups
- Creating and Managing Profile Sharing Groups
- Creating and Managing Service Groups
- Creating and Managing Customer Segments

20

Managing Account and Bill Unit Hierarchies

Learn how to create account and bill unit hierarchies in Oracle Communications Billing and Revenue Management (BRM) to organize accounts into groups for billing purposes or to represent the relationships between the accounts graphically in your CRM application.

Topics in this document:

About Account Groups

Accounts can be organized into groups for billing purposes or to represent the relationships between the accounts graphically in Billing Care or Customer Center. There are several kinds of account groups:

- **Hierarchical:** An account hierarchy has a parent account and any number of child accounts and other account hierarchies. Account hierarchies are created for the following reasons:
  - Solely to display relationships between accounts. In such cases, the parent account pays *none* of the child account's charges.
  - To provide a mechanism that custom reports can use to identify related accounts and analyze those relationships.
  - To provide the organizational structure for rolling charges of child accounts up to the parent account during billing.

> **Note:**
>
> Because bill unit hierarchies do not have to match account hierarchies, you do not need to set up account hierarchies to roll up charges among accounts. Typically, however, bill unit hierarchies exist within the structure of an account hierarchy, and applications such as Billing Care facilitate the creation of paying and nonpaying bill unit relationships during account hierarchy setup.
>
> For more information, see "About Bill Unit Hierarchies".

A child account can belong to only one parent account.

See "Managing Account and Bill Unit Hierarchies".

- **Charge or Discount Sharing:** A charge or discount sharing group consists of an owner account or service and one or more member accounts or services. Charge or discount sharing groups are created so that charges and discounts can be shared among accounts.

  See "Managing Charge and Discount Sharing Groups".

- **Sponsored top-up:** A sponsored top-up group consists of an owner account and one or more member accounts. The owner account can top up a specified balance in each member account. See "About Sponsored Top-Ups" in *BRM Configuring and Collecting Payments*.

In all types of account groups, the bills and payments belong to the account's bill units. See "About Bill Units and Account Groups".

About Bill Units and Account Groups

Accounts can have one or more bill units. Each bill unit stores billing information and tracks the charges for a particular bill. When accounts are set up with group relationships, the payment type of the bill units in the accounts, not the accounts themselves, determine which accounts pay the charges.

In account groups, bill units have an additional internal group structure:

- In an *account hierarchy*, the bill unit hierarchy can, but does not have to, match the account hierarchy. If necessary, a bill unit hierarchy can span multiple account hierarchies (that is, it can contain bill units of accounts that belong to different account hierarchies).

  Typically, however, an account hierarchy has a top-level parent bill unit and any number of child bill units and other bill unit hierarchies. The charges of the hierarchy's nonpaying child bill units are rolled up to the top-level paying parent bill unit during billing.

  If an account hierarchy is set up solely to display relationships among accounts, the bill unit in the parent account pays *none* of the charges in a child account's bill unit. Both the parent and child bill units are paying bill units.

  A child bill unit can belong to only one parent bill unit.

  See "Managing Account and Bill Unit Hierarchies".

- A *discount sharing group* or *charge sharing group* consists of a group owner and one or more members. The group owner's account has an owning balance group. The owning balance group bears the financial impact of the sharing group, and the bill unit for this balance group pays for the member bill unit's charges.

  Member accounts can have more than one bill unit. If a member account has multiple bill units, the bill unit that benefits from sharing is the one that has the balance group whose service has been chosen for sharing.

  See "Managing Charge and Discount Sharing Groups".

Comparing Hierarchy and Sharing

The main differences between hierarchies and sharing groups are:

Relationships

- **Hierarchy:** Creates parent-child relationships among accounts or bill units.
- **Sharing:** Creates owner/member relationships in which the owner assumes certain charges for the member or shares balances with the members. Discount sharing is based on discount offers purchased by the owner as part of charge offer charges. Charge sharing is based on chargeshare offers.

Bill Units and Balance Groups

- **Hierarchy:** Parent and child accounts have either paying bill units or nonpaying bill units.
- **Sharing:** Charge and discount sharing group owner accounts have owning balance groups that are part of bill units. For charge sharing, charges for eligible member events impact the owning balance group and associated bill unit. For discount sharing, shared discounts impact the member's balance groups by either increasing noncurrency balances or reducing the currency impact of an event.

Charge Offer and Service Ownership

- **Hierarchy:** Nonpaying child bill units of the same parent bill unit do not have to own the same charge offers.
- **Sharing:** An owner account does not need to have the same charge offers or services as the members.

  The owner bill units can provide sharing to group members even if, at group creation time, some members do not own the services for which sharing is provided. In this case, only members who currently own the shared services benefit from charge or discount sharing. Members who do not own the service when sharing is set up can participate in sharing if they purchase that service in the future.

Charge Offer Guidelines

- **Hierarchy:** A parent bill unit can pay for any charge offer.
- **Sharing:** An owner can pay for a member's charges in accordance with a chargeshare offer selected when the charge sharing group is created. An owner can share any discount included in the packages they purchase.

Number of Parents or Owners

- **Hierarchy:** A child account can belong to only one parent account, and a child bill unit can belong to only one parent bill unit.
- **Sharing:** Multiple owners can provide charge or discount sharing for the same service, and a member account can participate in multiple sharing groups for a service. Thus, a member bill unit can share charges with multiple owner bill units or receive discounts from multiple owners.

What the Parent or Owner Pays For

- **Hierarchy:** A parent bill unit pays all the charges for its nonpaying child bill units. The parent bill unit cannot pay for only some of a nonpaying child bill unit's charges.

  Bill items are created for nonpaying bill units, but the amounts in them are rolled up at billing to the paying bill unit's bill.
- **Sharing:** An owner bill unit pays only the member charges that are defined as shared when the charge sharing group is created.

  A charge sharing group can include some or all chargeshare offers in the database. The charges go directly to the bill items of the sharing group owner's bill unit.

Multiple Bill Units and Groups

- **Hierarchy:** If a child account has multiple bill units, the parent account can pay the charges for none, one, several, or all of the child account's bill units, depending on whether the bill units are paying or nonpaying.

- **Sharing:** If a member account has a service that participates in multiple charge and discount sharing groups from different owner accounts, the member's charges are distributed among the groups' owners and the member benefits from each owner's shared discounts. BRM distributes the charges among the owner bill units and applies discounts based on priorities defined when the member joins the sharing groups. The member account pays any charges that are not eligible for sharing.

Collecting Payment Information

- **Hierarchy:** During account creation, payment information is not collected for child accounts' nonpaying bill units. Their payment method is **Paid by Parent Account**.
- **Sharing:** During charge sharing group creation, the payment method of the owning balance group is set as the payment method for any shared charges. The members' payment method has no bearing on the sharing, so members can use any payment method.

  Discounts do not generate bills, they only change the amount of a bill. Therefore, payment method is not a factor for discount sharing groups.

Tracking Balance Impacts

- **Hierarchy:** Nonpaying bill units track and display their own balance impacts in real time. When you bill accounts in a hierarchy, balance impacts of nonpaying bill units are rolled up to their paying parent bill unit, and the account that owns the paying parent bill unit is billed for them. The account that owns the nonpaying bill unit and the account that owns the paying bill unit can be in different account hierarchies.
- **Sharing:** Member bill units do not track or display the balance impacts of shared charges. The owning balance group tracks the balance impacts of usage resulting from noncurrency shared discounts; for example, included minutes.

Balances

- **Hierarchy** Account hierarchies do not affect any type of balance. Bill unit hierarchies are concerned only with billing, so only currency balances are affected.
- **Sharing:** Charge and discount sharing groups affect all types of balances. For example, you can apply an owner's discount for included minutes to each member or apply a 15% discount on a member's monthly fee.

Hierarchy Balance Impacts

The parent-child relationship has no financial impact unless a child bill unit is a nonpaying bill unit. Nonpaying bill units maintain all their charges until billing, when the charges are rolled up to their paying parent bill unit, which is then responsible for the bill.

No financial impact occurs when you move an independent account or bill unit into a hierarchy and make it a child. If you make the bill unit nonpaying, you choose whether to bill it for bill-in-progress charges or to transfer the charges to the new paying parent bill unit.

When you move a nonpaying bill unit from one paying parent bill unit to another, you can specify which parent pays the bill-in-progress charges. By default, the bill-in-progress charges transfer to the new parent bill unit.

Before you can move a nonpaying bill unit out of a bill unit hierarchy, you must change it to a paying bill unit and specify whether the bill unit or its former parent will pay any bill-in-progress charges.

Sharing Group Balance Impacts

When a member account's service is added to a charge or discount sharing group, a financial relationship is created between the sharing group owner account and the member account.

- **Discount sharing:** If the owner account is active and the member account has services eligible for discount sharing, the shared discounts reduce the amount of money a member owes by applying the owner's discounts to the balance of the member account before finalizing the member's charges.
- **Charge sharing:** If the owner account is active and the member service has events eligible for charge sharing, the shared charges impact the balance of the owner account rather than the member account.

The member account stops benefiting from charge and discount sharing groups in the following situations:

- The owner account is inactive or closed.
- The sharing group is deleted from the owner account.
- The member account's service is removed from the sharing group.
- Events generated by a member fall outside of the validity period for the shared charge or discount.
- The charges or discounts that were included in the sharing group are deleted.
- For discount sharing groups, the noncurrency balances that were offered as part of the discount have been depleted in the owner account (for example, a monthly discount of included minutes that has already been consumed by the owner or other members). In this case, the member still benefits from currency-type shared discounts, such as a 10% reduction on bills for email usage or a $15 discount on overseas calls.

About Account Hierarchies

An account hierarchy is a set of accounts organized according to their positions in relation to each other. The relationships among accounts in a hierarchy are similar to parent-child relationships. The hierarchy is headed by a parent account with child accounts beneath it. At each level above the bottom of the hierarchy, the child accounts themselves can be parent accounts.

You set up account hierarchies for the following reasons:

- Solely to display relationships among accounts. In such cases, the parent account pays *none* of a child account's charges.
- To provide a mechanism that custom reports can use to identify related accounts and analyze those relationships.
- To provide the organizational structure for rolling charges of child accounts up to the parent account during billing.

> **Note:**
>
> Because bill unit hierarchies do not have to match account hierarchies, you do not need to set up account hierarchies to roll up charges among accounts. Typically, however, bill unit hierarchies exist within the structure of an account hierarchy, and applications such as Billing Care facilitate the creation of paying and nonpaying bill unit relationships during account hierarchy setup.
>
> For more information, see "About Bill Unit Hierarchies".

For example, a corporate customer might have several accounts for corporate employees, but the corporation itself pays all the employees' bills. In this case, the corporation's account is the parent account with the paying bill unit, and the employees' accounts are child accounts with nonpaying bill units.

An account's position in a hierarchy does not necessarily indicate whether it pays its own bills. The top-level parent account is not required to have a paying bill unit. Any account, either a parent or child, can have a paying bill unit or a nonpaying bill unit.

For more information about paying and nonpaying bill units in account hierarchies, see the following sections:

- About Bill Unit Hierarchies
- Calculating Balance Due in Account and Bill Unit Hierarchies

Performance Impact of Account Hierarchies

To maintain data consistency, many operations lock an account at the beginning of a transaction. Therefore, in an account hierarchy, many of the associated accounts are also locked. Although this provides reliable data consistency, it can cause a lot of serialization, which decreases the throughput of the system.

If this problem affects your system, you can choose to lock specific balance groups instead of the whole account.

> **Note:**
>
> Balance group locking might enable separate contexts to attempt to lock the same object, causing a system halt. When balance group locking is used, every feature that uses it should be examined for overlap.

For more information about balance group locking, see "Locking Specific Objects" in *BRM Developer's Guide*.

About Bill Unit Hierarchies

Bill unit hierarchies are a set of bill units organized according to their positions in relation to each other. A bill unit hierarchy is headed by a parent bill unit with child bill units beneath it. At each level above the bottom of the hierarchy, the child bill units themselves can be parent bill units.

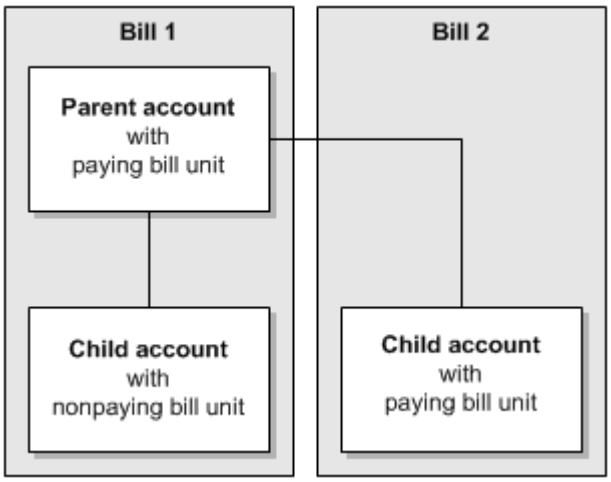
Typically, a bill unit hierarchy exists within the context of an account hierarchy used for billing purposes. The bill unit hierarchy can, but does not have to, match the account hierarchy. If necessary, a bill unit hierarchy can span multiple account hierarchies (that is, it can contain bill units of accounts that belong to different account hierarchies).

A bill unit's position in a hierarchy does not necessarily indicate whether it pays its own bills. Any bill unit, either a parent or child, can be a paying bill unit or a nonpaying bill unit, and both parent and child accounts can have paying bill units and nonpaying bill units:

- A paying bill unit pays its own bill and, if it is a parent, the bills of its nonpaying child bill units.
- A nonpaying bill unit's bill is paid by its parent bill unit. If its parent is also a nonpaying bill unit, its charges are rolled up the bill unit hierarchy to the next paying ancestor bill unit.

Figure 20-1 shows a simple bill unit hierarchy. The hierarchy contains three bill units, one in each account. Because the paying bill unit in the parent account pays the bill for the nonpaying bill unit in the child account, however, only two bills are generated for the three accounts.

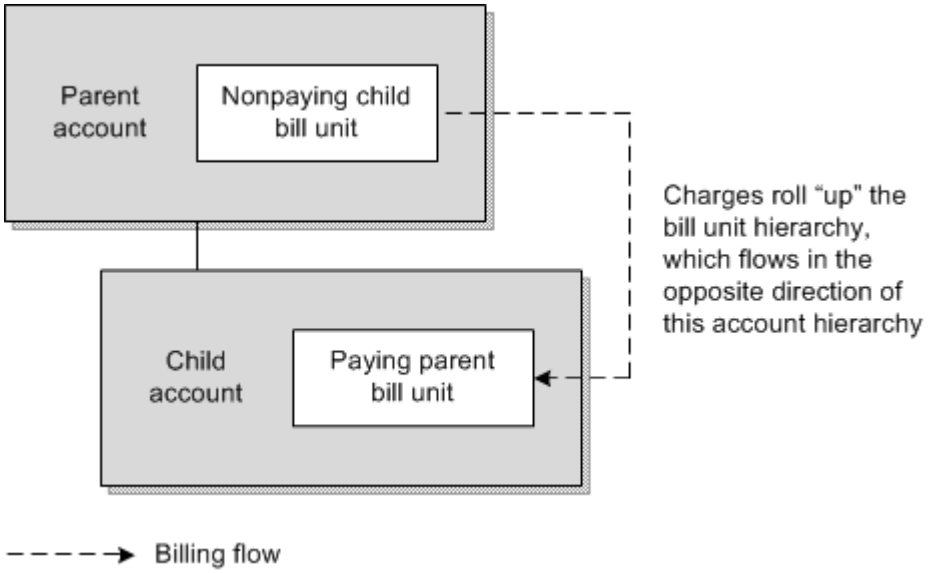
Figure 20-1 Simple Bill Unit Hierarchy



Account and bill unit hierarchies do not have to match. For example, a parent bill unit does not have to belong to a parent account, and a child bill unit does not have to belong to a child account.

Figure 20-2 shows an account hierarchy containing one parent account and one child account. Each account contains a bill unit. The *parent* account's bill unit is the nonpaying *child* of the *child* account's paying *parent* bill unit. In this situation, the child account pays the parent account's charges.

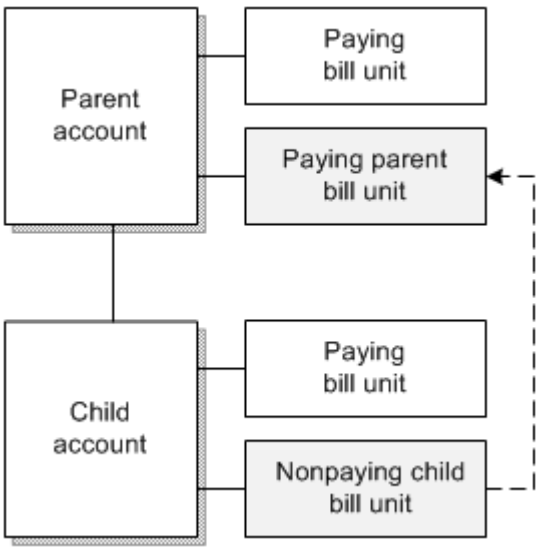
Figure 20-2 Nonmatching Account and Bill Unit Hierarchies



When hierarchical accounts have multiple bill units, the bill unit hierarchy becomes more complex. A child account can have both nonpaying bill units and paying bill units. The parent account is not required to pay all of the child account's bills.

Figure 20-3 shows a parent account and a child account with two bill units each. One of the child account's bill units is a paying bill unit. The charges for that bill unit are paid by the child account, not by the parent account. In this case, three bills are generated: one for the parent account, one for the parent-child account, and one for the child account.

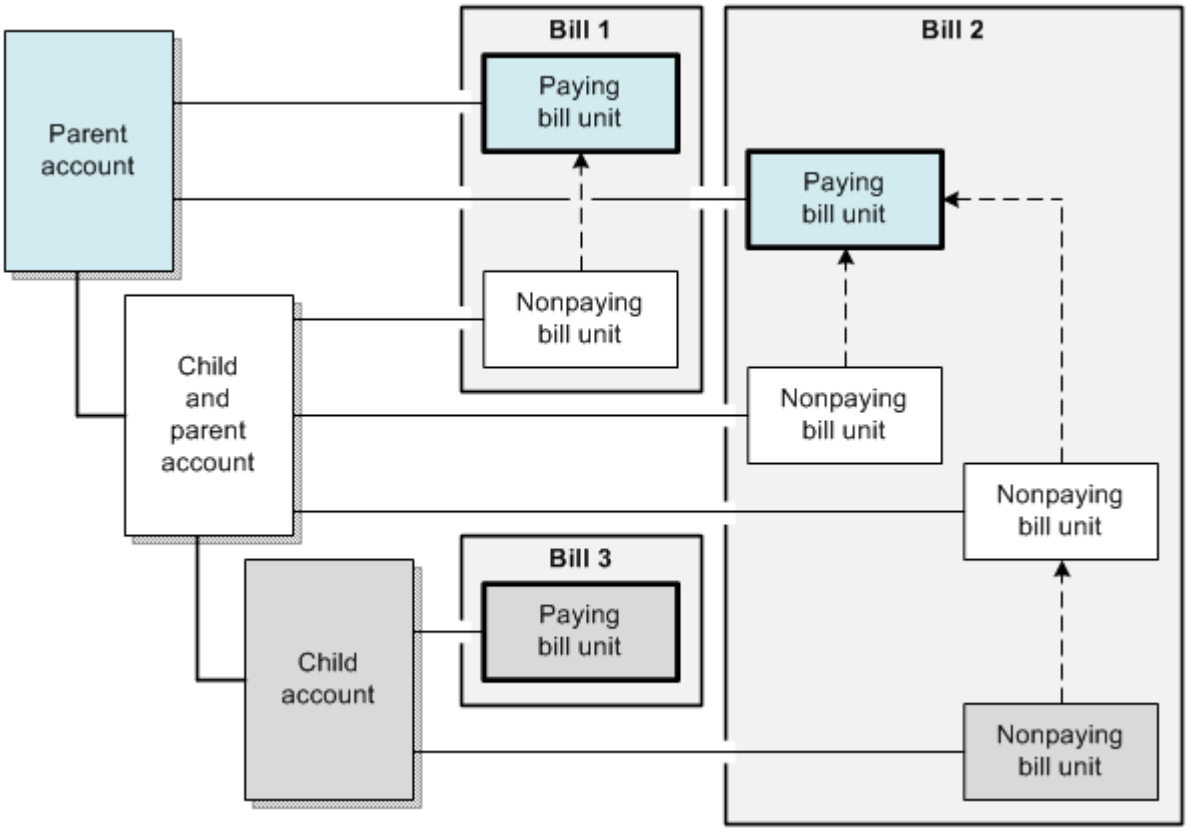
Figure 20-3 Complex Bill Unit Hierarchy



At each level above the bottom of a hierarchy, the child bill units themselves can be parent bill units.

Figure 20-4 shows a three-level account hierarchy containing seven bill units. Because only three of the seven bill units are paying, only three bills are generated. The top parent account receives two bills, and the bottom child account receives one bill.

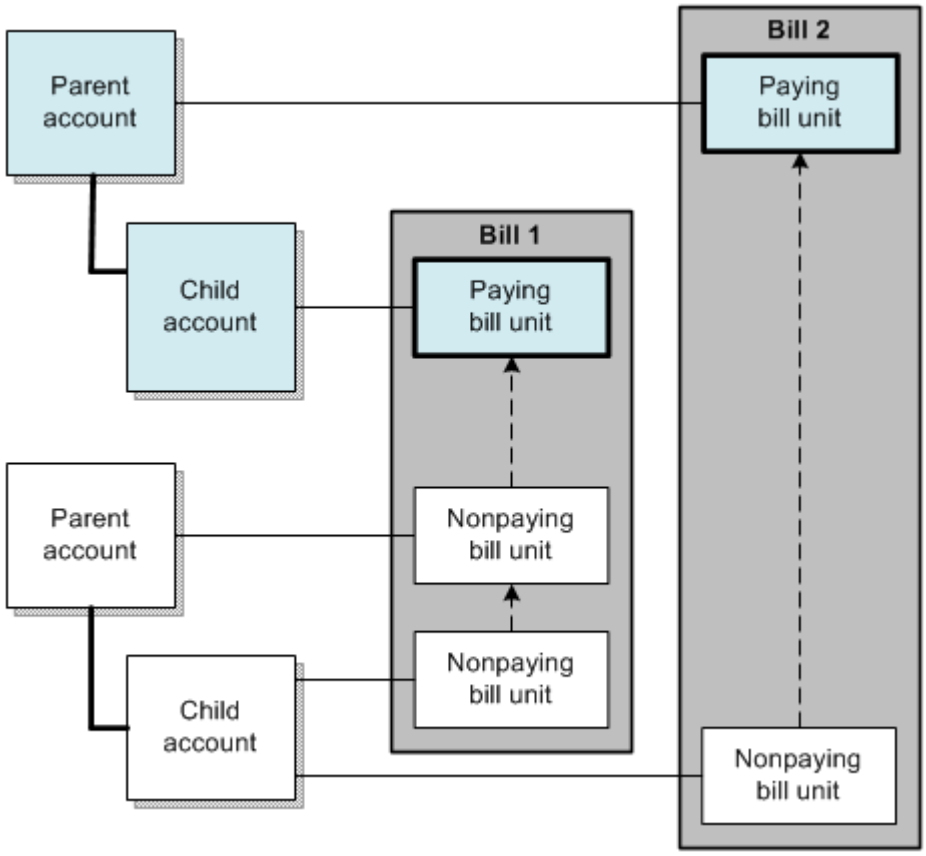
Figure 20-4 Three-Level Account Hierarchy with Multiple Bill Units



A bill unit hierarchy can contain bill units of accounts that belong to different account hierarchies.

Figure 20-5 shows two account hierarchies with a total of five bill units. The bill units are grouped into two bill unit hierarchies, each containing bill units from both account hierarchies. Because only two bill units are paying, only two bills are generated. One parent account receives a bill, and one child account receives a bill.

Figure 20-5 Bill Unit Hierarchies That Span Multiple Account Hierarchies



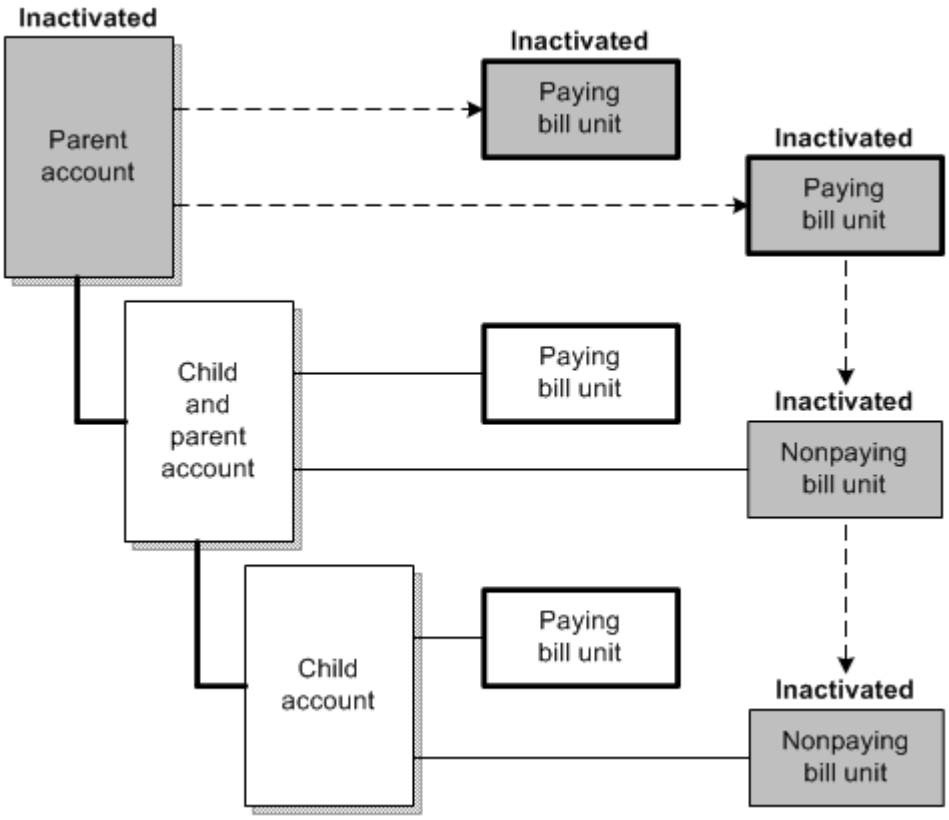
For information about creating bill unit hierarchies, see "Creating Account and Bill Unit Hierarchies".

How Account Status Changes Affect Hierarchies

Changing the status of an account in an account hierarchy changes the status of all accounts inside and outside the account hierarchy that have at least one nonpaying bill unit whose paying parent bill unit is owned by the initially changed account.

For example, Figure 20-6 shows the bill units that are inactivated in a multilevel hierarchy when a parent account is inactivated.

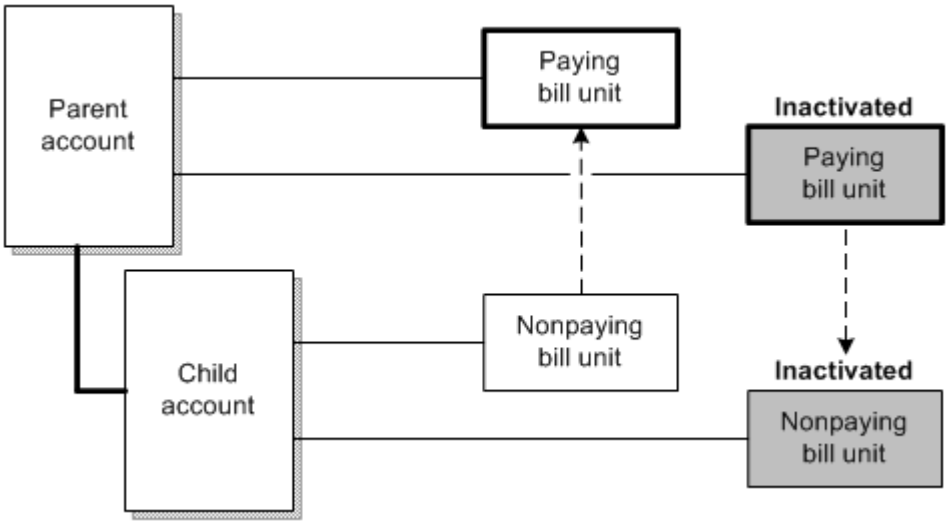
Figure 20-6 Effect of Parent Account Status Change on Bill Unit Hierarchy



Changing the status of a paying parent bill unit changes the status of all accounts that have at least one nonpaying child bill unit of the paying bill unit.

For example, Figure 20-7 shows the bill units that are inactivated in a multilevel hierarchy when the status of an individual paying bill unit is inactivated.

Figure 20-7 Effect of Parent Bill Unit Status Change on Bill Unit Hierarchy



Currency Requirements of Hierarchies

Nonpaying bill units must have the same currency as the account that owns their parent bill unit. If the accounts that own the parent and child bill units use two currencies, their primary and secondary currencies must match.

Billing Setups in Hierarchies

Because paying bill units handle the billing for nonpaying bill units, nonpaying bill units must have the same billing day of month (DOM), billing frequency, accounting cycle, and language as their paying parent bill unit.

Calculating Balance Due in Account and Bill Unit Hierarchies

An account hierarchy is a set of accounts organized according to their positions in relation to one another. Each hierarchy consists of one parent account and any number of child accounts and other account hierarchies.

When the accounts in a hierarchy are billed, however, the relationships among the accounts' bill units, not among the accounts, determine whether an account pays its own charges or rolls them up to another account inside or outside the hierarchy. To keep the charges within the account hierarchy, each bill unit in the hierarchy's accounts should be part of a bill unit hierarchy that is contained within the account hierarchy.

> **Note:**
>
> Bill unit hierarchies can, but do not have to, match account hierarchies. See "About Bill Unit Hierarchies".

Child accounts can have a paying bill unit or a nonpaying bill unit. The balance due is billed differently for accounts that have paying bill units and nonpaying bill units:

- The balance due for a paying bill unit is billed to itself.
- The balance due for a nonpaying bill unit is billed to its paying bill unit (the nonpaying bill unit's first paying bill unit ancestor).

BRM creates bills and bill items for all account bill units: parent, child, paying, and nonpaying. Billing, however, involves two different processes:

- Changing the status of bills and bill items to open, and creating new pending bills and bill items for the next bill. This occurs for all accounts.

  > **Note:**
  >
  > If a nonpaying child bill unit is billed before its paying parent bill unit, the nonpaying bill unit's items remains pending until the paying bill unit has been billed.

- Requesting a payment for the bill (for example, initiating a credit card transaction). This occurs for paying bill units only. A nonpaying bill unit never receives a payment request.

Each bill unit includes a pending bill and one or more pending bill items. As an account incurs balance impacts, such as usage fees, the balance due accumulates in the pending bill items.

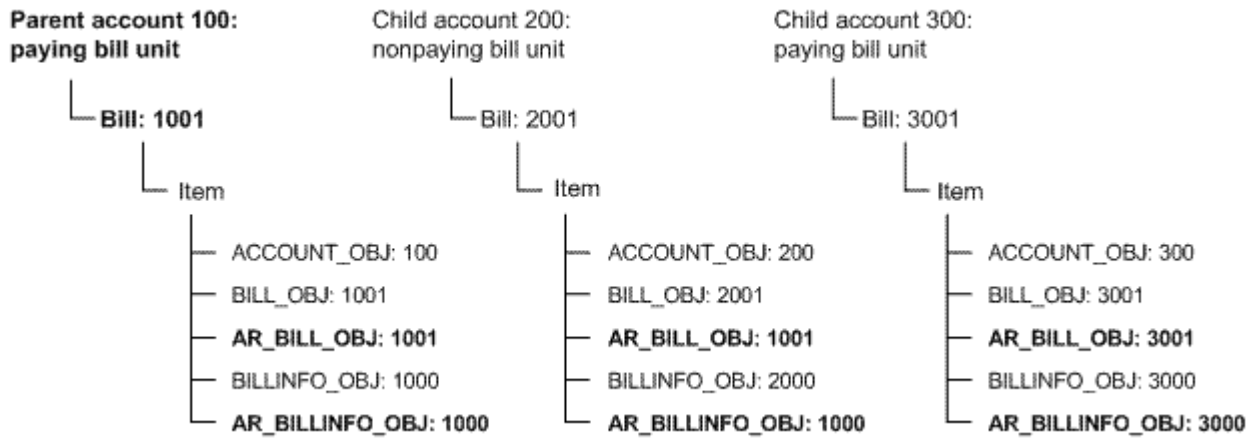
Each bill item includes these fields:

- **PIN_FLD_ACCOUNT_OBJ:** The account that the item belongs to. This field always points to the account that owns the item.
- **PIN_FLD_BAL_GRP_OBJ:** The balance group that the item belongs to.
- **PIN_FLD_BILLINFO_OBJ:** The bill unit that the item belongs to.
- **PIN_FLD_BILL_OBJ:** The bill that the item belongs to. This field always points to the account's own bill.
- **PIN_FLD_AR_BILLINFO_OBJ:** The paying bill unit that the item belongs to.
- **PIN_FLD_AR_BILL_OBJ:** The paying bill that the item belongs to.

The AR_BILLINFO_OBJ and AR_BILL_OBJ fields determine which account, bill unit, and bill handles billing for the item. If an item belongs to a nonpaying bill unit, the item's AR_BILLINFO_OBJ field points to the nonpaying bill unit's paying bill unit, and the AR_BILL_OBJ field points to a bill in the parent account of the paying bill unit. In a bill unit hierarchy with more than two levels, the balance due belongs to a nonpaying bill unit's first paying bill unit ancestor and its corresponding bill.

Figure 20-8 shows a parent account, a child account with a nonpaying bill unit, and a child account with a paying bill unit. Notice that in the child account with the nonpaying bill unit, the paying bill field (AR_BILL_OBJ) and paying bill unit field (AR_BILLINFO_OBJ) point to the parent account.

Figure 20-8 Paying Bill Unit Differences for Child Accounts



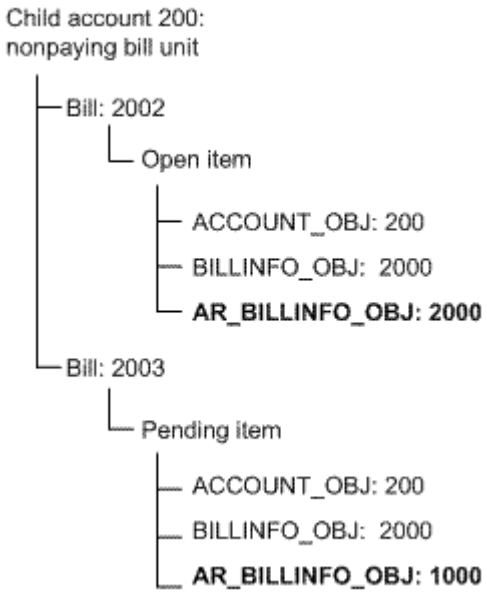
Who Pays for Open Items and Pending Items?

An account might have *open* items and *pending* items. This might happen if the customer has not paid an open bill or has paid only part of it. In that case, there would be an open item and a pending item.

If that account's bill unit changes from nonpaying to paying or from paying to nonpaying, the payment responsibility for only *pending* items is affected. Payment responsibility for *open* items is not changed.

Figure 20-9 shows a child account with a nonpaying bill unit that was a parent account with a paying bill unit, so it has a pending item and an open item. The open item points to the child account as the owner of the paying bill unit (AR_BILLINFO_OBJ), but the pending item points to the parent account as the owner of the paying bill unit.

Figure 20-9 Account with Pending and Open Items



In some cases, an account can accumulate balance impacts for part of a billing cycle before its bill unit becomes nonpaying. When the bill unit becomes a nonpaying child, the paying parent bill unit becomes responsible for pending charges accumulated before the child bill unit became nonpaying.

Figure 20-9 shows an account that is billed on the 5th day of each month. The account's bill unit becomes a nonpaying child on the 10th day of the month, but because it has already incurred balance impacts recorded in a pending item, the paying parent bill unit is billed for the balance due accumulated from the 5th through the 15th.

Figure 20-10 Account Charges Before Bill Unit Became Nonpaying

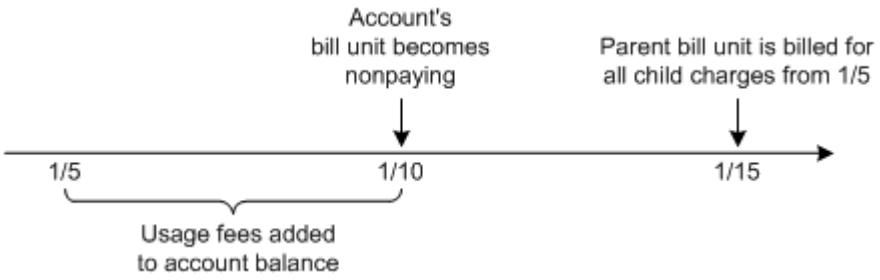


Table 20-1 summarizes changes to payment responsibilities.

Table 20-1 Effect of Paying/Nonpaying Changes on Account Hierarchies

Change to Account	Open Items	Pending Items
Parent account with paying bill unit becomes child account with nonpaying bill unit.	Nonpaying bill unit in the child account is responsible for its open item balance due.	Paying bill unit in the parent account is responsible for the pending item balance due of the child account's nonpaying bill unit.
Nonpaying bill unit in a child account becomes a paying bill unit.	Paying bill unit in the parent account is responsible for the open item balance due of the former child account's nonpaying bill unit.	Paying bill unit in the child account is responsible for its pending item balance due.
Child account with nonpaying bill unit changes parent account and paying parent bill unit.	Paying bill unit in the old parent account is responsible for the open item balance due of the child account's nonpaying bill unit.	Paying bill unit in the new parent account is responsible for the pending item balance due of the child account's nonpaying bill unit.

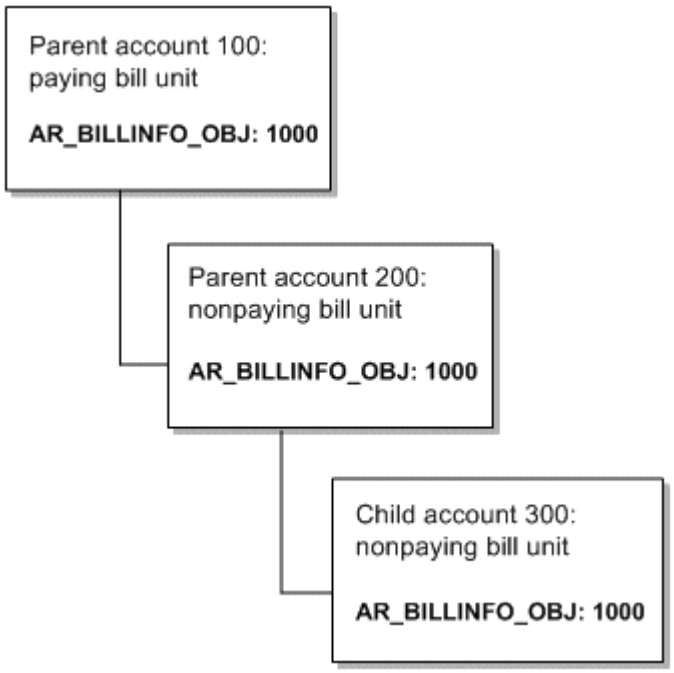
To close all of an account's open items before you make the account a nonpaying child, use the Bill Now feature in Billing Care or Customer Center.

Multiple Levels of Parent Accounts

If all child accounts (some of which might also be parent accounts) in an account hierarchy have only nonpaying bill units, the balance due for pending items is always handled by the hierarchy's top parent account.

In Figure 20-11, account 300, which has a nonpaying bill unit, is a child of account 200, which is in turn a child of account 100. The balance due for account 300 is handled by account 100, not by account 200.

Figure 20-11 Multiple Levels of Parent Accounts



Adding Child Due Amount to Intermediate Parent

You can control whether to add a child account's due, adjusted, disputed, written off, transferred, and received amounts from a child account to the corresponding amounts for a non-paying intermediate parent account in a hierarchical setup in BRM.

To add a child's due amount to its non-paying intermediate parent:

1. Go to *BRM_home*/**sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

3. Open the **bus_params_billing.xml.out** file.
4. In the XML file, add the below entry:

   ```
   <AddChildDueInIntermediateParentBill>value</
   AddChildDueInIntermediateParentBill>
   ```

   where *value* is one of the following:

   - **1**: The child's amount is added to its non-paying intermediate parent's amount.
   - **0**: The child's amount *is not* added to its non-paying intermediate parent's amount. This is the default.
5. Save the file as **bus_params_billing.xml**.
6. Load the XML file into the BRM database:

   ```
   pin_bus_params -v bus_params_billing.xml
   ```

7. Stop and restart the Connection Manager (CM).

Hierarchy Changes and Billing Dates

When you make an account a child account, BRM changes the child account's billing day of month (DOM) to match the parent account's billing DOM.

If the child account has a nonpaying bill unit whose paying parent bill unit is in the new parent account, the nonpaying bill unit must have the same billing DOM as the paying parent bill unit.

> **Note:**
>
> All bill units in a bill unit hierarchy must have the same billing DOM. BRM, however, does not enforce that requirement. Therefore, if the billing DOMs do not match in a bill unit hierarchy, you must manually align them.

The next billing date for the nonpaying bill unit, however, might not be the same as the next billing date of the paying parent bill unit. This happens because a change to a billing DOM always takes effect after the end of the current cycle.

As a result, the parent account's paying bill unit might be billed, but the nonpaying bill unit in the child account might not be billed. Therefore, the balance due for the nonpaying bill unit in the child account is not included in the parent account bill.

For example:

1. You create Account A on June 8.

   The billing DOM is 8.

2. You create Account B on June 20 and then change its billing day to 8.

   The next billing date is August 8. This happens because the current billing cycle, from June 20 through July 20, must be completed before the billing DOM changes. A long billing cycle, from June 20 through August 8, is created.

3. You make Account B a child account of Account A, changing Account B's bill unit to a nonpaying child of Account A's bill unit.

4. On July 8, Account A is billed, but Account B is not because Account B's next billing date is August 8.

   Therefore, the bill includes the balance due for only Account A, the parent account.

5. On August 8, both accounts are billed.

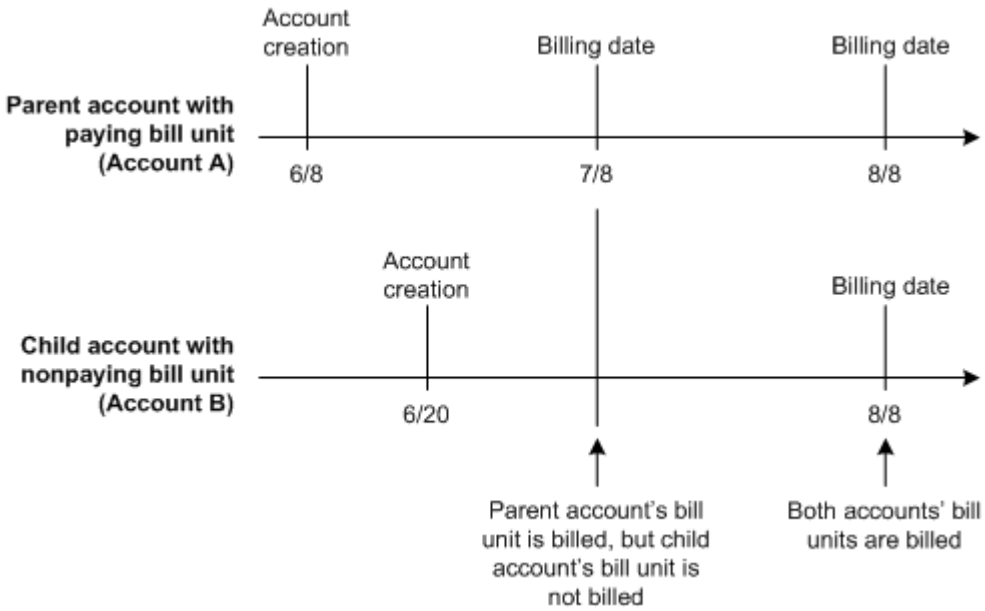
   The bill includes the balance due from both accounts:

   - Parent account balance due from July 8 to August 8.
   - Child account balance due from June 20 to August 8.

6. On all subsequent billing dates, both accounts are billed.

Figure 20-12 shows a timeline for the billing dates for two accounts in a parent-child relationship where the child account has a nonpaying bill unit.

Figure 20-12 Hierarchy Changes and Billing Dates



Examples of Changes to Account Hierarchies

These examples show who is responsible for the balance due following various account hierarchy changes.

> **Note:**
>
> Except where noted, these examples assume that the child account's nonpaying bill unit is owned by the parent account's paying bill unit.

An Account Becomes a Child Account with a Paying Bill Unit

In this case, there is no change to how the balance due is handled. The parent account handles its own balance due; the child account handles its own balance due.

An Account Becomes a Child Account with a Nonpaying Bill Unit When It Is Created

In this case, the total balance due is handled by the parent account, which owns the paying bill unit, and the billing date of the child account is the same as the parent account.

An Account Becomes a Child Account with a Nonpaying Bill Unit Immediately after It Is Created

In this case, the account has not been billed yet, so all its items are pending. Therefore, its total balance due is handled by the parent account, which owns the paying bill unit.

> **Note:**
>
> By default, for customers who pay by credit card, BRM charges purchase fees and the first cycle forward fee at account creation. The balance due for these fees is stored in open items and is charged to the child account's bill unit. You can turn off credit card collection at account creation by editing the **CCcollect** business parameter.

An Account Becomes a Child Account with a Nonpaying Bill Unit Several Months after It Is Created

In this case, the balance due for all pending items in the child account's nonpaying bill unit is handled by the parent account, which owns the paying bill unit. Any open items are handled by the child account. The billing date of the nonpaying bill unit in the child account is changed to match the billing date of the account that owns the paying bill unit.

A Child Account with a Nonpaying Bill Unit Changes Parent Accounts

> **Note:**
>
> This example assumes that the nonpaying bill unit's paying parent bill unit was also changed to a paying bill unit owned by the new parent account.

In this case, the balance due for all pending items in the nonpaying bill unit of the child account is now handled by a different parent account and that parent account's paying bill unit. Any open items are the responsibility of the former parent account. The billing date of the child account is changed to match the billing date of the new parent account, which owns the current paying parent bill unit.

A Child Account with a Nonpaying Bill Unit Becomes a Child Account with a Paying Bill Unit

In this case, the fields in pending items that specify the paying bill unit and the bill both point to the paying bill unit in the child account. If the parent account's bill unit that formerly paid the child's charges has any open items that include amounts due from the child account, the parent account's bill unit is responsible for those amounts even when the child account's bill unit is no longer nonpaying.

Creating Account and Bill Unit Hierarchies

You can create account and bill unit hierarchies by using Billing Care or Customer Center, or by using custom applications that call BRM opcodes.

By default, accounts are created with one bill unit. When you create account hierarchies, the bill units are automatically assigned the same hierarchical position as the accounts to which they belong.

To enable customers to receive separate bills for different services, you can create additional bill units per account. You then specify whether the bill units are paying or nonpaying and parents or children.

The bill units of accounts that do not belong to the same account hierarchy can form a bill unit hierarchy. Nonpaying child bill units in one account hierarchy can have a paying parent bill unit in a different account hierarchy.

The default validation in BRM doesn’t allow a non-paying bill unit without a paying parent. This can lead to validation errors when attempting to add a non-paying bill unit under the same account. To add a non-paying bill unit, you can use the PCM_OP_CUST_POL_VALID_BILLINFO opcode to override the default logic. This opcode allows you to customize the validation rules for bill unit hierarchies, enabling you to create non-paying bill units under the same account. See "Validating Bill Unit Data" in *BRM Opcode Guide* for more information.

Managing Account Hierarchies

To manage account hierarchies, you can do the following:

- Display account hierarchies in Billing Care or Customer Center.

  You can see the structure of an account hierarchy in the **Hierarchy** tab of the hierarchy's parent account. Initially, this tab shows the direct lineage of an account, not its siblings or the siblings of its parent.

- Change the parent of an account.

  You can use Billing Care or Customer Center to change an account's parent at any time. You change an account's parent by adding the account to a hierarchy, removing it from a hierarchy, or moving it from one hierarchy to another.

- Defer account hierarchy changes until a later date.

You can schedule a parent change for a future date. You can then use a daily billing utility, **pin_deferred_act**, to process the change automatically on the scheduled date.

See "Scheduling Status Changes in Advance".

Moving Closed Accounts into or out of Hierarchies

By default, BRM does not enable you to move closed accounts into or out of account hierarchies. To enable that functionality, configure the **allow_move_close_acct** entry in the CM configuration file.

1. Open the CM configuration file (*BRM_home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **allow_move_close_acct** entry to **1**:

   ```
   - fm_bill allow_move_close_acct 1
   ```

   By default, this entry is set to **0**.
3. Save and close the file.

21

Managing Charge and Discount Sharing Groups

Learn how to create charge and discount sharing groups in Oracle Communications Billing and Revenue Management (BRM) system to allow a group of people to all share the same charge or discount.

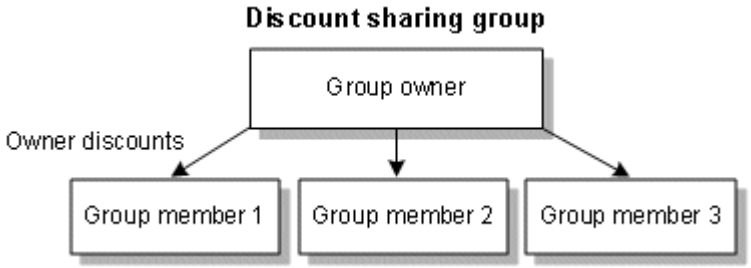
Topics in this document:

About Charge and Discount Sharing Groups

Accounts or services can share charges or discounts by joining groups that consist of an owner account or service and one or more member services:

- **Discount sharing group.** In a discount sharing group, the owner shares its discounts with the members, as shown in Figure 21-1.

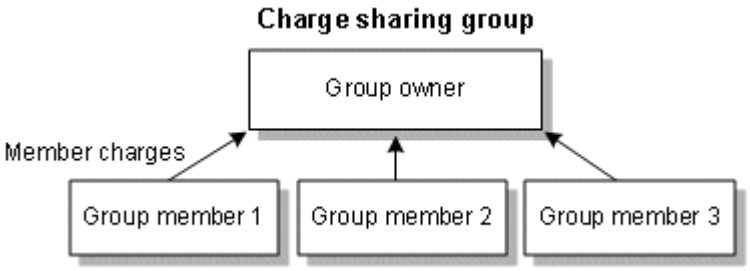
Figure 21-1 Discount Sharing Group



For more information, see "About Discount Sharing Groups".

- **Charge sharing group.** In a charge sharing group, the owner assumes charges that are incurred by the members, as shown in Figure 21-2.

Figure 21-2 Charge Sharing Group



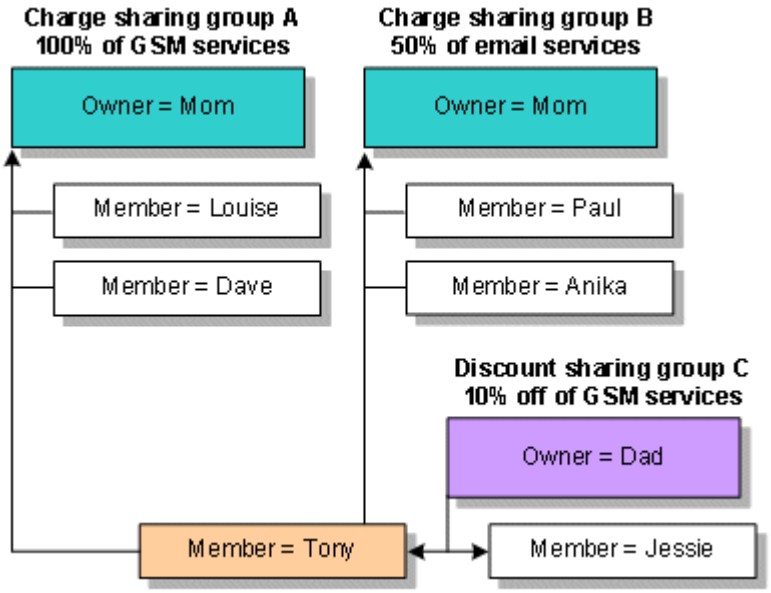
For more information, see "About Charge Sharing Groups".

You can use Billing Care or Customer Center to set up charge and discount sharing groups, or you can customize a third-party client application to set up sharing. For information on customizing charge and discount sharing, see *BRM Opcode Guide*.

Working with Complex Charge and Discount Sharing Groups

Networks of charge and discount sharing groups can be complex. An owner can have more than one charge or discount sharing group, and a member can belong to more than one charge or discount sharing group, as shown in Figure 21-3.

Figure 21-3 Complex Charge and Discount Sharing Groups



In this example, Mom owns two charge sharing groups: one for GSM services and one for email services. She assumes charges for all her children: 100% of GSM charges for Louise and Dave; 50% of email charges for Paul and Anika; and both GSM and email charges for her youngest son, Tony. Dad owns a discount sharing group; he shares a 10% discount on GSM charges with his son Tony and brother Jessie. The members have the following benefits:

- When Paul and Anika use email, Mom assumes part of their charges. They have no discounts, shared or owned, so Mom pays 50% of their monthly email charges and they pay the remaining 50%.

- When Louise and Dave use GSM services, Mom assumes their charges. They have no discounts, so Mom pays 100% of their monthly GSM charges.
- When Jessie uses GSM services, he receives a 10% discount from Dad. Jessie is not a member of any charge sharing groups, so he pays the remaining 90% of his GSM charges.
- When Tony uses email, Mom assumes part of his charges. He does not have any email discounts, so Mom pays 50% of his monthly email charges and he pays the remaining 50%.
- When Tony uses GSM services, he receives a 10% discount from Dad. This discount is applied before the charges are applied. Because Tony is a member of charge sharing group A, Mom pays the remaining 90% of his GSM charges.

In more complicated networks, an owner of a charge or discount sharing group can also be a member of another charge or discount sharing group. For example, Mom owns a charge sharing group that assumes email charges for Paul, Anika, and Tony. Mom, in turn, is a member of Grandma's charge sharing group, which also assumes email charges for its members. In this case, Grandma pays the email charges for Mom. In addition, she *inherits* the email charges for Paul, Anika, and Tony as a result of Mom's charge sharing group.

> **Note:**
>
> When setting up charge and discount sharing groups, avoid circular relationships. See "About Creating Charge and Discount Sharing Groups".

About Discount Sharing Groups

Discount sharing occurs when an account or service shares its discounts with other accounts' services. The account that shares its discounts is the owner of a discount sharing group. In the owner's account, one of the balance groups serves as the owning balance group. The owning balance group is determined based on whether the discount sharing group owner is the account or a service in the account:

- If the group owner is the account, the owning balance group is the account's default balance group, and all discounts purchased by the account are shared.
- If the group owner is a service in the account, the owning balance group is the associated service balance group, and only the discounts for that service are shared.

When a discount sharing group is created, a discount sharing group object (**/group/sharing/discounts**) is added to the database. This object contains the owner account or service, the list of discounts that are shared by the owner, and the list of members. The members are the services that use the shared discounts.

When the discount sharing group owner is a service, its service type must match the type of service to which the shared discounts apply.

When events are generated by member activity, discounts belonging to the sharing group owner are typically applied first, followed by any discounts belonging to the member account. After discounting, any remaining balance impact is applied to the member account unless the account also participates in charge sharing.

> **Note:**
>
> You can change the order in which discounts are consumed so that discounts belonging to the member are used before shared discounts.

For members to receive discounts from the owner, the member's ordered balance group must include a reference to the discount sharing group. This reference is created when the member joins the group. See "How Charges and Discounts Are Applied".

To create a discount sharing group, see "About Creating Charge and Discount Sharing Groups".

How Account Status Changes Affect Discount Sharing Groups

When the status of a discount sharing group's owner account is changed to inactive or closed, members stop benefiting from the shared discounts as follows:

- **Owner account is deactivated**: The status of all the group's shared discounts is changed to inactive. Members resume benefitting from the shared discounts when the owner's account is reactivated.
- **Owner account is closed**: All the group's shared discounts are removed from the account and from the discount sharing group.

> **Note:**
>
> If the **KeepCancelledProductsOrDiscounts** business parameter is set to **enabled**, the shared discounts are not removed from the discount sharing group. Instead, their status is set to canceled.

How Group Owner Changes Affect Discount Sharing Groups

If you use a third-party client application, you can use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT opcode to customize the application to enable a customer service representative (CSR) to change the owner of a discount sharing group. See *BRM Opcode Guide*.

When the owner of a discount sharing group changes, the discounts shared by the previous owner are deleted from the group, and discounts shared by the new owner are added.

Member-generated events, including delayed events, that occur after a group owner is changed impact the new owner's shared balances.

Members and Discount Sharing Groups

When you set up a discount sharing group, the members of the group must be services (the service balance groups).

If the owner of a shared discount is a service, the discount sharing group member service is typically the same service type as, or a subclass of, the discount owner service. For example, if the discount sharing group owner is **/service/telco**, the discount sharing group member might be **/service/telco/gsm**. This parent and subclass relationship is the standard way you set up

subscription services to offer a discount based on the total usage of all subscription services in an account group.

If the owner of a shared discount is a service, BRM requires that the owner and member services be of the same type. You can specify different service types for the owner and members by implementing the relevant subscription opcodes in your custom code (see *BRM Opcode Guide*). However, if the owner and member services are different types, the member service must match one of the shared discount's permitted service types (specified in the PIN_FLD_PERMITTEDS array in the **/discount** object).

If the owner and a member of the discount sharing group have different service types, to apply the discount to the member service, the member's usage event type must match the event type to which the shared discount applies.

For example, you set up a subscription discount that applies to **/service/telco** usage. In the Discount Attributes dialog box, you map the discount to the monthly cycle forward event. You then create a discount sharing group in which you share the telco discount that you created. You add an account to the discount sharing group and specify the account's **/service/telephony** service as the member. In this case, the subscription discount is applied to the member account only when the member account generates a monthly cycle forward event for its telephony service.

When you set up a discount sharing group, you can specify that the member service be a service type or a service instance:

- **Specify a service type as the member**: When you specify a service type (for example, GSM) as a member, the events generated by all service instances in that type are considered for discount sharing. Adding members by service type is appropriate in the following cases:
  - You add multiple member accounts to the group to take advantage of discounts for a service type (for example, an aunt who wants to share her GSM discounts with her nieces and nephews). In this case, each niece and nephew has a different service instance, so instead of specifying each service instance individually, you specify the member accounts and the service type. This captures all of the nieces' and nephews' service instances under the umbrella of the GSM service type.
  - You want a member account that has not yet purchased a service of that type to be automatically eligible for participation in discount sharing if the member buys the service in the future. With future sharing, even though a member becomes automatically eligible, you must track the member's purchases and, when the service is purchased, manually intervene to join discount sharing for the new service instance.
- **Specify a service instance as the member**: When you specify a service instance (for example, an employee's work phone) as a member, only the events generated by that instance are eligible for discounting. Do this to exclude other services of that type from discount sharing.

  For example, John's account includes work email (smith@CompanyA.com) and home email (john@internetprovider.com). John's employer wants to share a 10% discount for monthly fees incurred only by work email. Because John's account has both home and work services, you designate the service instance for smith@CompanyA.com as the member to prevent discount usage by john@internetprovider.com.

Currency Requirements of Discount Sharing Groups

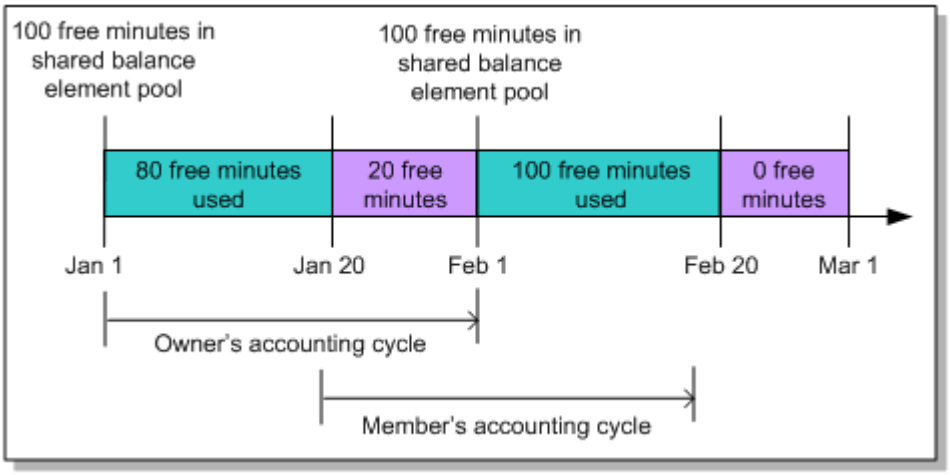
The discount sharing group's owner account and member accounts must use the same currency. If the accounts use two currencies, they must use the same primary currency.

Billing for Discount Sharing Groups

A discount sharing group's owner account and member accounts can have different accounting and billing cycles. Member-generated events that qualify for discounts always consume the available shared balances.

For example, in Figure 21-4, the owner's accounting and billing cycles begin on the 1st of the month, and the member's accounting and billing cycles begin on the 15th of the month. The owner receives 100 free monthly minutes that are shared.

Figure 21-4 Discount Sharing Between Accounts with Different Accounting Cycles



When a member generates an event that lasts 30 minutes on January 20, it consumes the available 20 minutes from the shared balance group. The member's account receives a balance impact for the remaining 10 minutes of the event.

When another member generates an event that lasts 15 minutes on February 20, there are no available minutes in the shared balance group to consume, and the member account receives a balance impact for the entire 15 minutes.

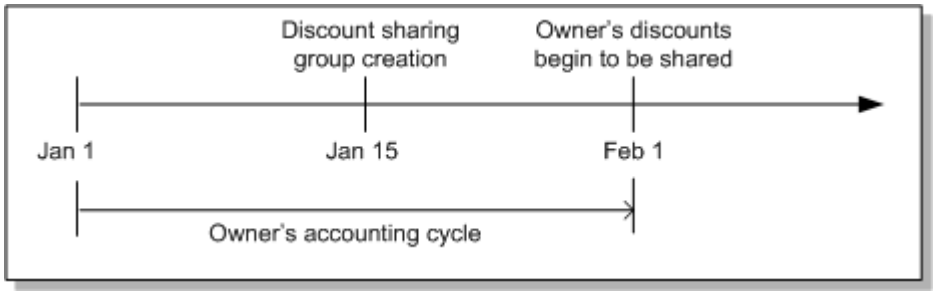
Configuring the Start and End Times for Discount Sharing

By default, discount sharing start and end times are based on the next billing cycle date. Therefore:

- When you create a discount sharing group, add a discount to an existing group, or add a member to the group, discount sharing starts at the beginning of the owner's next billing cycle.
- When you delete a discount sharing group, delete a discount from a group, or remove a member from the group, discount sharing is effective up to the end of the owner's billing cycle.

For example, if you create a discount sharing group on January 15 but the next billing cycle does not start until February 1, the members of the group do not receive any of the owner's shared discounts until February 1, as shown in Figure 21-5.

Figure 21-5 Effective Date of Discount Sharing Group Created Mid-Cycle



Conversely, if you delete a discount sharing group on January 15 but the next billing cycle does not start until February 1, the members of the group receive shared discounts based on the entire month of January, not just the first 15 days.

You can configure BRM to start and end discount sharing as soon as the group is created or deleted, a discount is added or deleted, or a member is added or removed. When you configure discount sharing this way, BRM prorates the shared discounts according to where the start or end date falls within the billing cycle. For example, if you delete a discount sharing group on April 15 and the owner's billing cycle starts on the first day of each month, the members receive 50% of the discount, equivalent to the portion of the month in which discount sharing existed.

To change the discount sharing start time:

1. Go to *BRM_home*/**sys/data/config**.
2. Use the following command to create an editable XML file from the **Subscription** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates an XML file named **bus_params_subscription.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_subscription.xml.out**, set **PropagateDiscount** to **enabled**:

   ```
   <PropagateDiscount>enabled</PropagateDiscount>
   ```

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. Use care when updating parameters in the file.

4. Save and exit the file.
5. Rename the **bus_params_subscription.xml.out** file to **bus_params_subscription.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

You do not need to restart the CM to update this entry.

About Charge Sharing Groups

Charge sharing enables a customer to sponsor the charges of other accounts or services in the system. For example, it enables a company to pay for all of its employees' GSM telephony services or a parent to pay for his child's SMS and email services.

You set up charge sharing by creating a charge sharing group, which consists of the following:

- **Charge sharing group owner**: The account or service responsible for all or a portion of the charges incurred by the charge sharing group members.
- **Charge sharing group members**: The accounts and services that the owner sponsors.
- **Chargeshare offer**: The chargeshare offer specifies how and when to apply charges. It is linked to a particular event type. Chargeshare offers determine whether events qualify for charge sharing and apply rules to calculate charge sharing amounts.

When a member incurs charges sponsored by the owner, the charges are applied to the owner's balance group first. Any charges that remain afterward impact the member's balance group.

Information about charge sharing groups is stored in **/group/sharing/charges** objects in the BRM database.

To create a charge sharing group, see "About Creating Charge and Discount Sharing Groups".

About Charge Sharing Group Owners

The account that sponsors the charges is the *owner* of a charge sharing group. Within the owner's account, one of the balance groups serves as the owning balance group. The owning balance group is the one that receives the balance impact of any shared charges incurred by the members.

The owning balance group is determined based on whether the charge sharing group owner is the account or a service in the account:

- If the group owner is the account, the owning balance group is the account's default balance group.
- If the group owner is a service in the account, the owning balance group is the associated service balance group.

How Owner Account Status Changes Affect Charge Sharing

By default, changing the status of a charge sharing group's owner account changes the status of all its shared charges. When the owner account is deactivated or closed, members stop benefiting from the charge sharing group. The owner no longer assumes any of the charges incurred by the members. Any new events generated by the members are charged to the member's balance groups *only*.

The way that BRM treats the charge sharing group depends on whether the owner account is deactivated or closed:

- **Owner account is deactivated**: All of the group's sponsored charges are suspended, and no new member charges are added to the owning balance group. Member services can begin benefiting from charge sharing again as soon as the owner's account is reactivated.
- **Owner account is closed**: All of the group's shared charges are suspended, and no new member charges are added to the owning balance group. The shared charges are removed from the charge sharing group.

> **Note:**
>
> If **KeepCancelledProductsOrDiscounts** is set to **1** in the Connection Manager (CM) **business parameters** file, the shared charges are not removed from the charge sharing group. Instead, their status is set to canceled.

About Changing Charge Sharing Group Owners

If you use a third-party client application, you can use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT opcode to customize the application to enable a CSR to change the owner of a charge sharing group. See *BRM Opcode Guide*.

When the owner of a charge sharing group changes, the charges that were sponsored by the previous owner are deleted from the group, and charges sponsored by the new owner are added.

Member-generated events, including delayed events, that occur after a group owner is changed impact the new owner's balances.

About Charge Sharing Group Members

When you set up a charge sharing group, the members of the group must be services (the service balance groups).

If the owner of a charge sharing group is a service, the group member service is typically the same service type as, or a subclass of, the charge sharing owner service. If the owner and a member of the charge sharing group have different service types, to share charges, the member's usage event type must match the event type to which the chargeshare offer applies.

> **Note:**
>
> BRM requires that the owner and member services be of the same type. You can specify different service types for the owner and members by implementing the relevant subscription opcodes in your custom code. See *BRM Opcode Guide*.

When you set up a charge sharing group and specify the member services, you can:

- **Specify a service type for a group of accounts as members**: When you specify multiple account instances and a service type (for example, Account 1, Account 2, and GSM) as members, only the service instances for each account are considered members. Adding members by service type is appropriate when:

  - You are adding multiple member accounts to the group and you want them all to take advantage of charge sharing for a service type (for example, an aunt who wants to pay the GSM charges for all of her nieces and nephews). In this case, each niece and nephew has a different service instance, so instead of specifying each service instance individually, you specify the member accounts and the service type. This captures all of the nieces' and nephews' service instances under the umbrella of the GSM service type.
  - You want a member account that has not yet purchased a service of that type to be automatically eligible for participation in charge sharing if the member buys the service in the future. With future sharing, even though a member becomes automatically eligible, you must track the member's purchases and, when the service is purchased, manually intervene to join charge sharing for the new service instance.

- **Specify a service instance as the member**: When you specify a service instance (for example, an employee's work phone) as a member, only the events generated by that instance are eligible for charge sharing. Specify membership by using a specific service instance to exclude other services of that type from charge sharing.

  For example, John's account includes work email (smith@CompanyA.com) and home email (john@internetprovider.com). John's employer wants to pay for monthly fees incurred by work email only. Because John's account has both home and work services, you designate the service instance for smith@CompanyA.com as the member to prevent the employer from paying for usage charges from john@internetprovider.com.

- **Specify all accounts in the system as members**: When you specify a type-only account POID as the member, events generated by any account in your system are eligible for charge sharing.

  See "About Global Charge Sharing Groups".

- **Specify all services of a specific type as members**: When you specify a type-only service POID as the member, events generated for that service type are eligible for charge sharing.

  See "About Global Charge Sharing Groups".

For member charges to be applied to the owner's account, the member's ordered balance group must include a reference to the charge sharing group. This reference is created when the member joins the group. For more information, see "How Charges and Discounts Are Applied".

> **Note:**
>
> This requirement does not apply to global charge sharing groups. See "About Global Charge Sharing Groups".

Currency Requirements for a Charge Sharing Group

The charge sharing group's owner account and member accounts must use the same currency or primary currency.

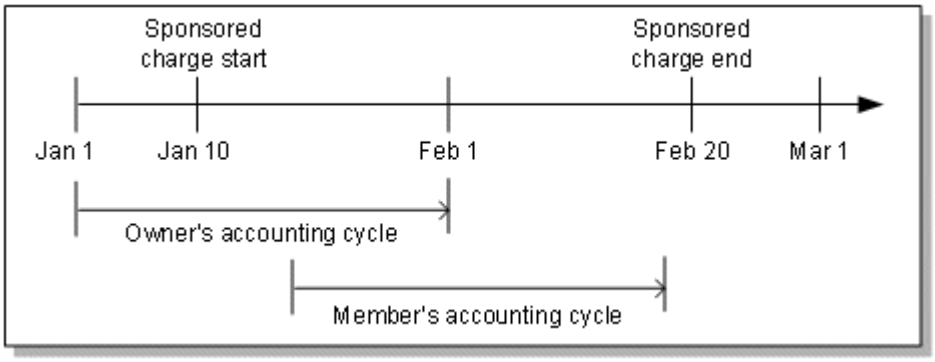
> **Note:**
>
> The currency requirement does not apply for global charge sharing groups. See "About Global Charge Sharing Groups".

Billing for a Charge Sharing Group

A charge sharing group's owner and member accounts can have different accounting and billing cycles. Sponsored charges are always applied to the owner account, regardless of when the member-generated events occur.

In the example illustrated in Figure 21-6, the owner's accounting and billing cycles begin on the 1st of the month, and the member's accounting and billing cycles begin on the 15th. Charge sharing starts on January 10 and ends on February 20, so between those dates, the owner sponsors 50% of all charges incurred by a member.

Figure 21-6 Effective Dates of Charge Sharing Groups Created Mid-Cycle



If the member generates an event on January 20 for 30 minutes at $0.10 per minute (total cost $3.00):

- The owner's account receives a balance impact of $1.50, and the amount appears on the owner's bill on February 1.
- The member's account receives a balance impact of $1.50, and the amount appears on the member's bill on February 15.

The member receives the balance impacts for all charges incurred after February 20.

About Global Charge Sharing Groups

> **Note:**
>
> Global charge sharing groups are not supported in ECE-based deployments.

You can create a charge sharing group that includes all accounts in your system or all services of a specific type, such as GSM telephony. This type of charge sharing group is called a global

charge sharing group. This enables one account to pay for all or a portion of the charges incurred by anyone in your system.

You specify which events the group owner pays for and how to split charges between the owner and members by creating a chargeshare offer.

For example, you can configure BRM to charge a company for all or a portion of any call to a toll free number or any text message sent to a specified number.

Global charge sharing groups differ from charge sharing groups in these ways:

- Global charge sharing groups are not listed in a member's ordered balance group. Therefore, the order in which charges are applied to global charge sharing groups is set by the system.
- Members in the global charge sharing group are indicated by a type-only POID.

About the Order in Which Charges Are Applied to Groups

The order in which BRM applies charges to the different types of charge and discount sharing groups is built into the system. BRM automatically applies charges to global charge sharing groups after applying any discounts from discount sharing groups but before applying charges to any charge sharing groups. The order in which BRM applies charges and discounts is shown below:

1. Discount sharing groups as listed in the member's ordered balance group
2. Global charge sharing groups having type-only services
3. Global charge sharing groups having type-only accounts
4. Charge sharing groups as listed in the member's ordered balance group

> **Note:**
>
> Only charge sharing groups and discount sharing groups are listed in a member's ordered balance group. Global charge sharing groups are not included in the list.

About Creating Global Charge Sharing Groups

Global charge sharing groups, like charge sharing groups, consist of an owner, members, and the chargeshare offer that defines when and how to apply charges. However, instead of listing every group member, the global charge sharing group indicates all accounts in the system or all services of a specific type as members by using a type-only POID.

> **Note:**
>
> Charge sharing group members must be either a type-only POID or a list of member POIDs; members cannot include both types.

Any existing account or service in your system is automatically considered a part of the group. Likewise, when you create a new account or service in the system, BRM automatically considers the account or service as part of the global charge sharing group.

To create global charge sharing groups, perform the following:

1. Make sure the global charge sharing search is enabled. See "Enabling Global Charge Sharing Searches During Discounting".
2. Use Billing Care, Customer Center, or a custom client application to create global charge sharing groups. To create global charge sharing groups by using a custom client application, see "Using Third-Party Applications to Manage Global Charge Sharing Groups".

Enabling Global Charge Sharing Searches During Discounting

By default, the global charge sharing search is disabled.

- With this feature *enabled*, BRM searches for global charge sharing objects during the discounting process.
- With this feature *disabled*, rating does not search for global charge sharing objects during the discounting process, thereby resulting in better performance.

To enable this feature, run the **pin_bus_params** utility to change the **EnableGlobalChargeSharing** business parameter. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To enable global charge sharing searches during discounting:

1. Go to *BRM_home***/sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

3. In the XML file, change **disabled** to **enabled**:

   ```
   <EnableGlobalChargeSharing>enabled</EnableGlobalChargeSharing>
   ```

4. Save the file as **bus_params_rating.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_rating.xml
   ```

6. Stop and restart the CM.

Using Third-Party Applications to Manage Global Charge Sharing Groups

To set up a third-party client application to create, modify, and delete global charge sharing groups, you must customize the application to accept the following information and to pass it in the input flist to the Subscription Management FM opcodes:

- Name of the global charge sharing group
- Owner of the group
- Members of the group
- The chargeshare offer associated with the group

To add all accounts in the system as members, you pass a type-only account POID and a NULL service POID in the input flist's PIN_FLD_MEMBERS array. For example:

```
0 PIN_FLD_MEMBERS           ARRAY [0] allocated 2, used 2
1   PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account -1 0
1   PIN_FLD_SERVICE_OBJ      POID [0] 0.0.0.0 /service 0 0
```

To add all services of a specific type in the system as members, you pass a type-only account POID and a type-only service POID in the input flist's PIN_FLD_MEMBERS array. For example:

```
0 PIN_FLD_MEMBERS            ARRAY [0] allocated 2, used 2
1    PIN_FLD_ACCOUNT_OBJ      POID [0] 0.0.0.1 /account -1 0
1    PIN_FLD_SERVICE_OBJ      POID [0] 0.0.0.1 /service/email -1 0
```

> **Note:**
>
> In both cases, you pass a type-only account POID in the PIN_FLD_ACCOUNT_OBJ field. The field cannot be NULL.

You pass the information from your client application to the following Subscription Management FM opcodes:

- To create a global charge sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE.
- To modify a global charge sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY.
- To delete a global charge sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE.

> **Note:**
>
> For global charge sharing groups, *do not* call the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode.

For more information about these opcodes, see *BRM Opcode Guide*.

About Creating, Modifying, and Deleting Charge and Discount Sharing Groups

This section covers the following topics:

About Creating Charge and Discount Sharing Groups

To create a charge or discount sharing group, you specify the following:

- A group owner account or service
- Member services
- (Discount sharing) The group owner's discounts to share with the members

- (Charge sharing) The charges, or chargeshare offers, that the group owner assumes for the members

Be aware of the following restrictions:

- By default, all members of a sharing group must reside in the same database schema. To enable sharing groups to include members from multiple schemas, see "Enabling Group Members to Reside in Multiple Schemas".
- Charges or discounts selected for sharing must be valid. A valid charge or discount is one that has an active or inactive status and has not expired.
- The owner of a charge or discount sharing group cannot also belong to a member-owned charge or discount sharing group of the same type.

  For example, Anna is the owner of discount sharing group DG1 and wants Sam to be a member. But Sam is the owner of a different discount sharing group, DG2. Anna can have Sam as a member of DG1 *only* if she is not a member of DG2.
- The name of a charge or discount sharing group must be unique and cannot be used for another sharing group owned by the group owner.
- When adding a service as a member, you can add the service as either a service type or a specific service instance.

  If you add a service type as a member, all instances of that service type become members. However, the subtypes of the service type do not become members. For example, if you specify the GSM service type as a member, all instances of GSM become members, but GSM fax, GSM telephony, and so forth do not.
- All members must have the same primary currency as the owner.

  > **Note:**
  >
  > This restriction does not apply to global charge sharing groups.

After you create a charge or discount sharing group, members can join the group. When they join the group, an ordered balance group is created. The ordered balance group determines which sharing groups the member can use and the order in which the groups are used. See "How Charges and Discounts Are Applied".

You create charge and discount sharing groups in Billing Care or Customer Center.

To implement charge and discount sharing through a custom client application, use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode. Use the PCM_OP_SUBSCRIPTION_POL_GET_SPONSORS opcode to get a list of chargeshare offers. See *BRM Opcode Guide*.

When creating a discount sharing group, the moment when sharing starts is determined by a business parameter. See "Configuring the Start and End Times for Discount Sharing".

About Modifying Charge and Discount Sharing Groups

You can modify a charge or discount sharing group in the following ways:

- **Delete members from the group.**

  When a member is deleted from the group, the member can no longer use discounts from the shared balances or have its charges assumed by the owner. Deleting a member

removes the member from the group. It also removes the group from the member's ordered balance group list. See "How Charges and Discounts Are Applied".

- **Delete chargeshare offers or shared discount offers.**

  When a chargeshare offer is deleted, the owning balance group no longer receives balance impacts for member-generated events that are part of that chargeshare offer. When a shared discount offer is deleted, members can no longer apply that discount, and any owner balances for that discount cannot be used by the members.

  Deleting a chargeshare offer or a shared discount offer removes it from the group. It also removes the group from the members' ordered balance group lists.

- **Add members to the group.**

  When a member is added to the group, that member can begin benefiting from sharing by joining the group. Adding new members updates the group's members list.

  After you add members to the group, they can join the group. When they join the group, an ordered balance group is created and sharing for that member can begin. See "How Charges and Discounts Are Applied".

- **Add shared charges or discounts.**

  When charges are added to a charge sharing group, the owner account begins to receive the balance impact of those charges from the member services. When discounts are added to a discount sharing group, members can begin to use the discounts for the events generated by their accounts.

  Adding charges or discounts updates the group's sponsors or discounts list. The charges must be defined as chargeshare offers in the database, and the discounts must be owned by the group owner. The charges and discounts must have an active or inactive status and valid dates.

- **Change the owner of the group.**

  When the owner of a discount sharing group changes, members use discounts from the new owner's shared balances rather than from the old owner's balances. When the owner of a charge sharing group changes, the new owner assumes charges generated by the members, and the old owner stops assuming the charges.

  Changing the owner assigns a new group owner, deletes the sponsors or discounts list belonging to the previous owner, and creates a new sponsors or discounts list containing the charges or discounts shared by the new owner.

You modify charge and discount sharing groups in Billing Care or Customer Center.

If you are implementing discount sharing through a third-party client application, you customize the application to invoke BRM opcodes. See *BRM Opcode Guide* and About Deleting Charge and Discount Sharing Groups .

When members or discounts are added to or deleted from a discount sharing group, the moment when sharing starts and ends is determined by a business parameter. See "Configuring the Start and End Times for Discount Sharing".

About Deleting Charge and Discount Sharing Groups

A charge or discount sharing group is deleted when the group's owner account is closed or when the sharing group is deleted by a CSR. When a discount sharing group is deleted, members can no longer use the discounts that were shared by the owner. When a charge sharing group is deleted, the owner no longer assumes members' charges.

When a charge or discount sharing group is deleted, BRM also deletes the sharing group from each member's ordered balance group, effectively ending the membership. For more information about ordered balance groups, see "How Charges and Discounts Are Applied".

You delete charge and discount sharing groups in Billing Care or Customer Center.

If you are implementing charge or discount sharing through a third-party client application, use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE opcode. See *BRM Opcode Guide*.

When a discount sharing group is deleted, the moment when sharing ends is determined by a business parameter. See "Configuring the Start and End Times for Discount Sharing".

Enabling Group Members to Reside in Multiple Schemas

By default, all members of a sharing group must reside in the same database schema. If you want to include a member from a different schema, you must first use **pin_amt** to migrate the member to the same schema as all other sharing group members.

Alternatively, you can enable sharing groups to include members from other database schemas. To do so, use the **pin_bus_params** utility to modify the **CrossSchemaSharingGroup** business parameter.

To enable group members to reside in multiple schemas:

1. Use the following command to create an editable XML file from the **system** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```

   This command creates the XML file named **bus_params_system.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **CrossSchemaSharingGroup** to **enabled**:

   ```
   <CrossSchemaSharingGroup>enabled</CrossSchemaSharingGroup>
   ```

3. Save the file and change its name to **bus_params_system.xml**.

4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_system.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

6. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

How Charges and Discounts Are Applied

An account can purchase discounts, share discounts, and have charges that are sponsored. For events generated by a member service, the shared discounts are used before the sponsored charges are applied to the owning balance group. That way, sponsored charges are applied after usage charges have been discounted. To control the sequence in which shared charges and discounts are applied for member-generated events, BRM maintains an ordered balance group for each member.

About Ordered Balance Groups

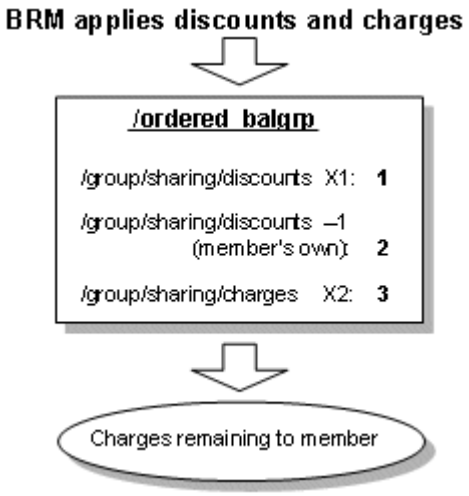
Each member of a charge or discount sharing group has an ordered balance group (**/ordered_balgrp** object). Ordered balance groups are created when members first join sharing groups and are updated each time members join new sharing groups or leave sharing groups.

An ordered balance group contains links to the groups that the member has joined, listed by rank. The ordered list determines the sequence in which the group's balances are impacted by the member-generated events.

> **Note:**
>
> Ordered balance groups do not include any global charge sharing groups to which the member belongs. For more information, see "About Global Charge Sharing Groups".

The following example, as illustrated in Figure 21-7, shows how BRM applies charges and discounts for a member's ordered balance group that references discount sharing group X1 and charge sharing group X2.

Figure 21-7 BRM Application of Discounts and Charges to Ordered Balance Group



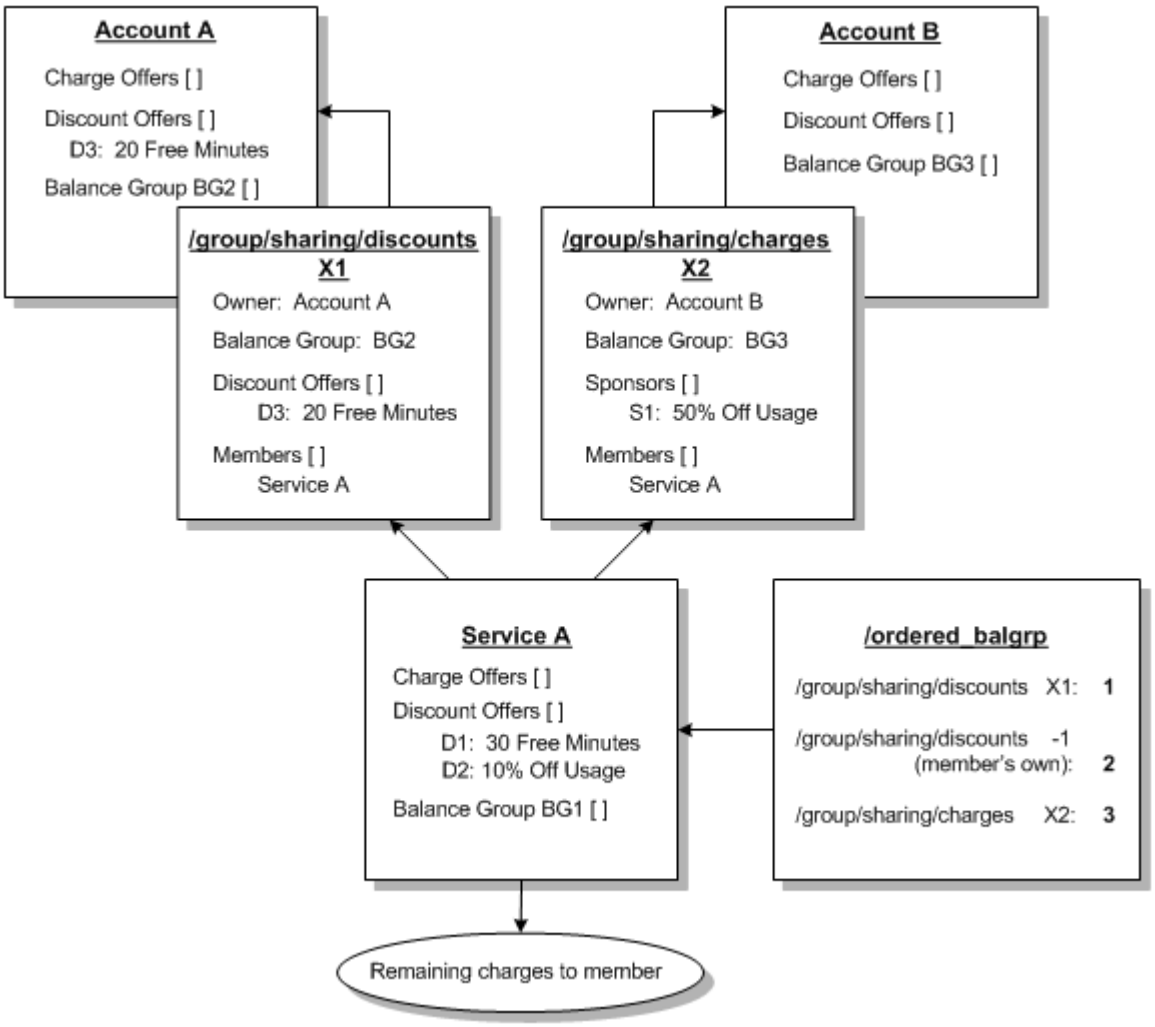
> **Note:**
>
> In this example, the entry indicated by **-1** references discounts owned by the member.

As shown in Figure 21-7, when the ordered balance group contains *both* charge and discount sharing groups, the discount sharing group's balances are used first. This enables discounts to be applied before usage charges are determined for charge sharing. If multiple charge or discount sharing groups are configured, they are applied in the order of their rank.

In addition, BRM *always* references any charges owed by the member after referencing the ordered balance group. That way, members receive charge sharing benefits before having to use their own charges.

The following example, illustrated in Figure 21-8, shows how a member's ordered balance group is used to impact the balances of the sharing groups. In this example, Account A is the owner of discount sharing group X1. Account B is the owner of charge sharing group X2. Service A is a member of both X1 and X2. Account A shares a discount that grants 20 minutes of telephone calls monthly. Account B sponsors 50% of usage charges. Service A has two discounts, one for 30 minutes monthly and another that discounts 10% of the charges.

Figure 21-8 Ordered Balance Group Impact on Charge and Discount Sharing Groups



Here, BRM applies shared charges and discounts for events generated by Service A as follows:

1. Service A generates an event for 100 minutes of telephone calls at $0.10 per minute.

   Before any discounts or sponsored charges are applied, Service A owes $10.00.

2. Account A's discounts are consumed first.

   After consuming 20 minutes, Service A owes $8.00 for 80 minutes of telephone calls at $0.10 per minute.

3. Service A's discounts are consumed next.

   After consuming 30 minutes, Service A owes $5.00 for 50 minutes of calls at $0.10 per minute. After the 10% usage discount, Service A owes $4.50 for 50 minutes of calls.

4. Account B's charges are consumed last.

   After a 50% of usage sponsored by Account B, Account B owes $2.25 and Service A owes $2.25.

How Discounts Are Applied When a Member Belongs to the Group Owner Account

When a member of a discount sharing group belongs to the discount sharing group owner account, the discount at the account level is applied twice for the member.

For example, Account A is the owner of discount sharing group X. Account A owns Service A and Service A is also a member of the discount sharing group X. Account A shares a discount that grants 20 minutes. When Service A generates an event for 100 minutes, the event is discounted as follows:

- Service A is granted 20 minutes as a member of the discount sharing group X.
- Because Service A also belongs to the account and any discount purchased at the account level applies to its services, the account discount is applied and Service A is granted an additional 20 minutes.

Creating, Deleting, and Modifying Ordered Balance Groups

Ordered balance groups are created for a member service that participates in charge or discount sharing. A member account can have multiple ordered balance groups depending on how many services are included in sharing groups. For example, if a member account participates in sharing for its IP, email, and GSM services, it has ordered balance groups for each of these services.

Creating Ordered Balance Groups

To create an ordered balance group, you specify the charge and discount sharing groups that provide shared balances for the service or account.

You create ordered balance groups when a member first joins charge or discount sharing groups for a service.

To implement charge or discount sharing through a third-party client application, use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See *BRM Opcode Guide*.

Deleting Ordered Balance Groups

To delete an ordered balance group, you specify the ordered balance group that you want to remove from the database.

To delete an ordered balance group when implementing charge or discount sharing through a third-party client application, use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See *BRM Opcode Guide*.

Modifying Ordered Balance Groups

Modifying an ordered balance group consists of adding or deleting charge or discount sharing groups:

- **Adding a sharing group**: A link to the charge or discount sharing group is added to the ordered balance group list.
- **Deleting a sharing group**: The link to the charge or discount sharing group is removed from the ordered balance group list.

> **Note:**
>
> If you add or delete charge or discount sharing groups, you might need to rearrange the ordered balance group list. See "Modifying the Order in Which Charge and Discount Sharing Groups Are Used ".

If you are implementing charge or discount sharing through a third-party client application, use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See *BRM Opcode Guide*.

Modifying the Order in Which Charge and Discount Sharing Groups Are Used

The sequence of the ordered balance group list determines which shared balances are used first. You can change the order of this list to reprioritize sharing for a member.

You modify the order of the ordered balance group list when you join charge and discount sharing groups or when a member decides to use one charge or discount sharing group before another.

If you are implementing charge or discount sharing through a third-party client application, use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See *BRM Opcode Guide*.

Creating or Modifying Multiple Ordered Balance Groups Simultaneously

To create or modify multiple ordered balance groups simultaneously, you specify the charge and discount sharing group that provides the shared balance group. You also specify a list of member services to share with this balance group.

If you are implementing charge or discount sharing through a third-party client application, use the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP_BULK_MODIFY opcode. See *BRM Opcode Guide*.

22

Managing Product Sharing Groups and Discount Offer Sharing Groups

Learn how to use product sharing groups to allow a group of customers to all own the same product in Oracle Communications Billing and Revenue Management (BRM). Discount sharing groups allow a group of customers to all own the same discount offer.

Topics in this document:

About Product Sharing

Product sharing enables your customers to share a product with a group of accounts. For example, a corporate account could purchase a Wireless product with special pricing, and then automatically share the package with all of its employees. In this case, the corporate account and all of its employees would own the same product and enjoy the same special pricing.

> **Note:**
>
> Product sharing is supported only by BRM systems that use Elastic Charging Engine (ECE) for usage charging.

About Discount Offer Sharing

Discount offer sharing enables your customers to share a discount offer with a group of accounts. For example, a corporate account could purchase an Internet discount offer that includes a 15% discount on the monthly fee and three free movie rentals, and then automatically share the discount offer with all of its employees. In this case, the corporate account and each of its employees would receive a 15% monthly fee discount and three free movie rentals.

> **Note:**
>
> Discount offer sharing is supported only by BRM systems that use ECE for usage charging.

About Sharing Groups

You specify the accounts that can share a product or discount offer by creating a sharing group. Each group includes the following:

- **The owner**. Owners have permission to purchase, modify, or remove the product or discount offer that is shared with members of the group. They can also add or remove group members.

  > **Note:**
  >
  > An owner can own only one product sharing group and only one discount offer sharing group at a time. Also, owners cannot be a member of another sharing group.

- **All members**. Members of a sharing group can include accounts and services.

> **Note:**
>
> By default, all members of a sharing group must reside in the same database schema. To enable sharing groups to include members from multiple schemas, see "Enabling Group Members to Reside in Multiple Schemas".

If you want BRM to automatically create sharing groups for you (called *automated sharing groups*), also define the group type. Group types specify the type of members in a hierarchy that the group can contain. For example, group membership can be restricted to nonpaying child accounts and their services, or it can include both paying and nonpaying child accounts and their services. Table 22-1 shows the group types supported by automated sharing groups.

Table 22-1 Supported Automated Sharing Group Types

Group Type	Description
Hierarchy	Includes paying and nonpaying child accounts and their services.
Payment Responsibility	Includes nonpaying child accounts and their services. In this case, the charges from all child accounts are rolled up to the group's owner.

About the Sharing Group Creation Process

You create and modify sharing groups by using a third-party customer relationship management (CRM) application. You must configure your CRM application to accept the following information for each product sharing group or discount offer sharing group, and then pass it to the BRM API:

- Sharing group name
- Sharing group owner
- List of group members
- (Automated Sharing Groups Only) The group type:

  - For product sharing: Hierarchy product sharing group (H_PSG) or payment responsibility product sharing group (PR_PSG)
  - For discount offer sharing: Hierarchy discount sharing group (H_DSG) or payment responsibility discount sharing group (PR_DSG)
- (Optional) The product or discount offer to share with all members of the group

The process that BRM uses to create or modify a sharing group depends on whether you implement it in your system with or without Automated Monitor Setup (AMS).

- **With AMS**, you can automate many of the group creation and modification processes. AMS automatically adds and removes members from the sharing group for you.
- **Without AMS**, you add and remove members manually to sharing groups.

BRM creates sharing groups as follows:

1. Takes as input all information about the sharing group.
2. Adds members to the sharing group. BRM determines whether the following is true:
   - AMS, automated sharing groups, and currency resource monitoring are enabled.
   - The group type is H_PSG, PR_PSG, H_DSG, or PR_DSG.

   If both are true, BRM uses AMS to add members. See "About Using AMS to Create Automated Sharing Groups".

   If either one is false, BRM adds members without AMS. See "About Manually Creating Sharing Groups".

About Using AMS to Create Automated Sharing Groups

You configure BRM to automatically add and remove members from sharing groups, based on the group type, by using AMS. AMS is a group of opcodes that automate many of the processes for creating and updating sharing groups.

AMS creates and updates automated sharing groups by following the account hierarchy or billing hierarchy associated with accounts and services. This synchronizes the membership between the hierarchy and the automated sharing group. AMS uses the owner and group type to find all appropriate child accounts and services in the hierarchy, and then adds them to the automated sharing group. When the owner purchases a product or discount offer, it automatically adds the product or discount offer to all members.

AMS automatically adds and removes members from sharing groups when the following occur:

- **A sharing group is created**. AMS takes as input the owner and the group type, and then searches through the owner's lineage to find accounts and services that match the criteria for the group type. All subordinate accounts and services in the hierarchy that meet the criteria are added as members.

  For example, when users create a payment responsibility-type sharing group, AMS adds to the group the owner's account and its services and all subordinate nonpaying child accounts and their services.
- **An account or billing hierarchy is changed**. AMS automatically adds or removes members from a sharing group, based on the group type. For example, when you add a nonpaying child account to a hierarchy, AMS automatically:
  - Finds all parent accounts at a higher level in the hierarchy than the specified child account.
  - Finds all automated sharing groups associated with each parent account.

  - Adds the child account and its services to each sharing group.

When an automated sharing group is created or a hierarchy changes, the following occurs:

1. BRM generates a notification event.
2. The BRM event notification system calls one of the AMS opcodes.
3. The AMS opcodes add or remove members from the automated sharing group and adds the product or discount offer to all member accounts.

To configure your system to create or update automated sharing groups, see "Setting Up Automated Sharing Groups".

About Manually Creating Sharing Groups

When creating or updating a product sharing group or a discount offer sharing group without AMS, you are responsible for adding and removing members manually. BRM does not verify that the groups include all appropriate child accounts and services in a hierarchy nor does it check whether the hierarchy changes over time.

You must manually add or remove members when the following occur:

- **A sharing group is created**. You specify the individual accounts and services to add to the sharing group.
- **An account or billing hierarchy is changed**. You must manually update the sharing group associated with the hierarchy. When an account is added to a hierarchy, you must manually add the account to the sharing group. Likewise, when an account is removed from a hierarchy, you must manually remove the account from the sharing group.

To configure your custom client application to manually create or update sharing groups, see "Setting Up Manual Sharing Groups in Custom Client Applications".

Setting Up Automated Sharing Groups

To set up BRM to create automated product sharing groups and automated discount offer sharing groups for you, do the following:

1. Enable AMS and automated sharing groups in BRM.

   See "Enabling Automated Sharing Groups in BRM".
2. Enable currency resource monitoring in BRM.

   See "Enabling Currency Resource Monitoring".
3. Specify the type of organization hierarchy that BRM synchronizes with for each group type.

   See "Mapping Organization Hierarchy Types to Automated Sharing Group Types".
4. (Optional) Configure sharing groups to allow members from other database schemas.

   See "Enabling Group Members to Reside in Multiple Schemas".
5. Configure BRM event notification to call the AMS opcodes.

   See "Configuring Event Notification for Automated Sharing Groups".
6. Configure your custom client application to call the appropriate API for managing automated sharing groups.

   See "Implementing Automated Sharing Groups in Custom Client Applications".

7. Create the products or discount offers that can be shared with members of an automated sharing group.
   - For automated product sharing groups, see "Creating Charge Offers with Automatic Sharing Enabled".
   - For automated discount offer sharing groups, see "Creating Discount Offers with Automatic Sharing Enabled".

After your system is set up, you can use your custom client application to create automated sharing groups and then have the group owner purchase one of the charge offers or discount offers created in step 6.

Enabling Automated Sharing Groups in BRM

To enable AMS and automated sharing groups in BRM, run the **pin_bus_params** utility to modify the following business parameters in the **subscription** instance of the **/config/business_params** object:

- **AutomatedGroupSharingSetup**: Enables automated sharing groups in BRM.
- **AutomatedMonitorSetup**: Enables AMS.

See "pin_bus_params" in *BRM Developer's Guide* for information about the utility's syntax and parameters.

To enable automated sharing groups:

1. Go to *BRM_home*/**sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r -c "Subscription" bus_params_subscription.xml
   ```

   This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.
3. Open the XML file.
4. Set **AutomatedGroupSharingSetup** to one of the following:
   - **0**: Disables both automated product sharing groups and automated discount offer sharing groups.
   - **1**: Enables automated discount offer sharing groups.
   - **2**: Enables automated product sharing groups.
   - **3**: Enables both automated product sharing groups and automated discount offer sharing groups.

   For example, this enables both automated product sharing groups and automated discount offer sharing groups:

   ```
   <AutomatedGroupSharingSetup>3</AutomatedGroupSharingSetup>
   ```
5. Set **AutomatedMonitorSetup** to **enabled**:

   ```
   <AutomatedMonitorSetup>enabled</AutomatedMonitorSetup>
   ```
6. Save the file and change its file name to **bus_params_subscription.xml**.
7. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

8. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Enabling Currency Resource Monitoring

Automated sharing groups rely upon the currency resource monitoring functionality included with the balance monitoring feature. Therefore, you must enable the balance monitoring feature. To do so, use the **pin_bus_params** utility to modify the **BalanceMonitoring** business parameter.

To enable currency resource monitoring:

1. Use the following command to create an editable XML file from the multibalance instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

   This command creates the XML file named **bus_params_multi_bal.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **BalanceMonitoring** to **enabled**:

   ```
   <BalanceMonitoring>enabled</BalanceMonitoring>
   ```

3. Save the file and change its name to **bus_params_multi_bal.xml**.

4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_multi_bal.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

6. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Mapping Organization Hierarchy Types to Automated Sharing Group Types

When creating or modifying an automated sharing group, BRM searches through an account's hierarchy lineage, based on the group type, to find the owner or members in the group. You specify the type of organization hierarchy to synchronize with an automated sharing group type by using business profiles. For automated sharing groups, create business profiles that map:

- Account hierarchies (**CorporateAutoSharingGroupType**) to hierarchy product sharing groups (**H_PSG**) and hierarchy discount sharing groups (**H_DSG**).
- Billing hierarchies (**PostpaidAutoSharingGroupType**) to payment responsibility product sharing groups (**PR_PSG**) and payment responsibility discount sharing groups (**PR_DSG**).

To map organization hierarchy types to automated sharing groups types:

1. Open the **pin_business_profile.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home*/**sys/data/config** directory.

2. In the **CorporateAutoSharingGroupType** business profile, set the **H_PSG** and **H_DSG** keys to enabled (**1**):

```
<BusinessProfilename="CorporateAutoSharingGroupType" >
   <Desc> Description of the business profile. </Desc>
   <!-- List key values -->
   <NameValue key="H_DSG" value="1"/>
   <NameValue key="PR_DSG" value="0"/>
   <NameValue key="H_PSG" value="1"/>
   <NameValue key="PR_PSG" value="0"/>
</BusinessProfile>
```

3. In the **PostpaidAutoSharingGroupType** business profile, set the **PR_PSG** and **PR_DSG** keys to enabled (**1**):

```
<BusinessProfilename="PostpaidAutoSharingGroupType" >
   <Desc> Description of the business profile. </Desc>
   <!-- List key values -->
   <NameValue key="H_DSG" value="0"/>
   <NameValue key="PR_DSG" value="1"/>
   <NameValue key="H_PSG" value="0"/>
   <NameValue key="PR_PSG" value="1"/>
</BusinessProfile>
```

4. Save and close the file.

5. Use this command to load the **pin_business_profile.xml** file into the database:

```
load_pin_business_profile pin_business_profile.xml
```

> **Note:**
>
> - When you run the utility, the **pin_business_profile.xml** and **business_configuration.xsd** files must be in the same directory. By default, both files are in *BRM_home*/**sys/data/config**.
> - This utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.
> - If you do not run the utility from the directory in which **pin_business_profile.xml** is located, include the complete path to the file.
>
> See "load_pin_business_profile" in *BRM Managing Customers*.

6. To verify that the business profile and validation template information was loaded, display the **/config/business_profile** objects and **/config/template** subclass objects by using one of the following features:

   - Object Browser
   - **robj** command with the **testnap** utility

   For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

7. Stop and restart the CM.

Enabling Group Members to Reside in Multiple Schemas

By default, all members of a sharing group must reside in the same database schema. If you want to include a member from a different schema, you must first use **pin_amt** to migrate the member to the same schema as all other sharing group members.

Alternatively, you can enable sharing groups to include members from other database schemas. To do so, use the **pin_bus_params** utility to modify the **CrossSchemaSharingGroup** business parameter in the **system** instance of the **/config/business_params** object.

To enable group members to reside in multiple schemas:

1. Use the following command to create an editable XML file from the **system** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```

   This command creates the XML file named **bus_params_system.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **CrossSchemaSharingGroup** to **enabled**:

   ```
   <CrossSchemaSharingGroup>enabled</CrossSchemaSharingGroup>
   ```

3. Save the file and change its name to **bus_params_system.xml**.

4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_system.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

6. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Configuring Event Notification for Automated Sharing Groups

To configure the BRM event notification system to call AMS opcodes when creating and maintaining automated sharing groups:

1. In your *BRM_home***/sys/data/config/pin_notify** file, ensure that the following lines in bold are uncommented:

   ```
   # Automated Monitor Setup related event notification
   # Uncomment these lines when AMS feature is enabled
   7854    0       /event/notification/service/pre_purchase
   7855    0       /event/notification/service/pre_purchase
   7856    0       /event/notification/service/pre_purchase
   ```

```
7857    0       /event/customer/billinfo/modify
7855    0       /event/customer/billinfo/modify
7857    0       /event/group/member
7854    0       /event/group/member
7857    0       /event/notification/bal_grp/modify
7855    0       /event/notification/bal_grp/modify
7854    0       /event/audit/subscription/transfer

# Automated Group Setup related event notification
# Uncomment these lines when feature is enabled
7853    0       /event/group/sharing/products/create
7853    0       /event/group/sharing/discounts/create
```

2. If your system has multiple configuration files for event notification, merge them.

   See "Merging Event Notification Lists" in *BRM Developer's Guide*.

3. Load your final **pin_notify** file into the BRM database.

   a. Go to the directory that contains your final **pin_notify** file.

   b. Run the following command to run the **load_pin_notify** utility:

      load_pin_notify *configFileName*

      where *configFileName* is the name of your final **pin_notify** file. If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file.

      See "load_pin_notify" in *BRM Managing Customers* for information about the utility's syntax and parameters.

   c. Stop and restart the Connection Manager (CM).

Implementing Automated Sharing Groups in Custom Client Applications

You can configure a custom client application to manage automated product sharing groups and automated discount offer sharing groups in BRM by calling either BRM opcodes or Billing Care REST API endpoints. Table 22-2 lists the API to call for managing automated sharing groups.

Table 22-2 Implementation Using BRM API

Action	BRM Opcode	Billing Care REST API Endpoint
Create product or discount offer sharing groups	PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE	**Create a Sharing Group**
Modify product or discount offer sharing groups	PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY	**Update a Sharing Group**
Delete product or discount offer sharing groups	PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE	**Delete a Sharing Group**

When a product sharing group or discount offer sharing group is:

- Created successfully, BRM generates an **/event/group/sharing/products/create** or **/event/group/sharing/discounts/create** event
- Modified successfully, BRM generates an **/event/group/sharing/products/modify** or **/event/group/sharing/discounts/modify** event

- Deleted successfully, BRM generates an **/event/group/sharing/products/delete** or **/event/group/sharing/discounts/delete** event

These events trigger the BRM event notification system to call the AMS opcodes, which automatically add and remove members from product sharing groups and discount offer sharing groups (**/group/sharing/products** or **/group/sharing/discounts** objects) and then update each member's **/ordered_balgrp** object.

For more information, see the following:

- For Billing Care REST API: See the **Sharing** REST endpoints in *REST API Reference for Billing Care*
- For opcodes: See "Managing Sharing Groups" in *BRM Opcode Guide*

Creating Charge Offers with Automatic Sharing Enabled

To be able to automatically share a product with members of an automated product sharing group, you must create charge offers in Pricing Design Center (PDC) with automatic sharing enabled.

The way you enable automatic sharing depends on the method you use to create your product offerings:

- Using the PDC UI. When creating a charge offer, select the **Share charge offer with members of billing hierarchy** check box. For more information, see "Specifying Charge Offer Settings" in *PDC Online Help*.
- Using the **ImportExportPricing** utility. When creating a charge offer in your XML file, set the **<groupSharingEnabled>** element under the **<chargeOffering>** element. See "Sharing Products Automatically with Group Members" in *PDC Creating Product Offerings*.

Creating Discount Offers with Automatic Sharing Enabled

To be able to automatically share a discount offer with members of an automated discount sharing group, you must create discount offers in PDC with automatic sharing enabled.

The way you enable automatic sharing depends on the method you use to create your product offerings:

- Using the PDC UI. When creating a discount offer, select the **Share discount offer with members of billing hierarchy** check box. For more information, see "Specify Discount Offer Settings" in *PDC Online Help*.
- Using the **ImportExportPricing** utility. When creating a discount offer in your XML file, set the **<groupSharingEnabled>** element under the **<alterationOffering>** element. See "Sharing Discount Offers Automatically with Group Members" in *PDC Creating Product Offerings*.

Setting Up Manual Sharing Groups in Custom Client Applications

To create or update sharing groups manually, you must configure your custom client application to do the following:

1. Create product sharing groups (**/group/sharing/products** objects) and discount offer sharing groups (**/group/sharing/discounts** objects). To do so, call one of the following:
   - The PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode. See "Creating a Product Sharing Group" in *BRM Opcode Guide* for more information.

- **Create a Sharing Group** operation in the Billing Care REST API. See the **Sharing** operations in *REST API Reference for Billing Care*.

2. Modify an existing product sharing group or discount offer sharing group by adding members or changing the owner. To do so, call one of the following:
   - The PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY opcode. See "Modifying a Sharing Group" in *BRM Opcode Guide* for more information.
   - **Modify a Sharing Group** operation in the Billing Care REST API. See the **Sharing** operations in *REST API Reference for Billing Care*.
3. Add product sharing groups and discount offer sharing groups to each member's list of sharing groups (in **/ordered_balgrp** objects). You must create or update the **/ordered_balgrp** object associated with each account and service that you want to include in a sharing group. To do so, call one of the following:
   - The PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode. See "Managing Ordered Balance Groups" in *BRM Opcode Guide* for more information.
   - **Update Shared Services for a Member** operation in the Billing Care REST API. See the **Sharing** operations in *REST API Reference for Billing Care*.

23

Managing Balance Monitoring Groups

Learn how to use balance monitoring to enable your customers to monitor the balance of a group of accounts in near real time in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Balance Monitoring

You use balance monitoring to enable your customers to monitor the balance of a group of accounts in near real time. Balance monitoring collects the balance impacts for a specified group of accounts and services, and notifies customers when their balance is too high. Customers and CSRs can also access the group's total balance at any time by using a custom client interface.

For example, a family with multiple wireless telephony accounts can create a balance monitor that tracks the entire family's balance and that alerts them when the balance total exceeds $100.

Balance monitoring provides advantages to:

- *Service providers* because it reduces the risk of debt exposure from nonpaying customers. Customers who track their balances are less likely to accrue excessive charges that they cannot pay.
- *Customers* because they can monitor spending habits and are alerted when charges exceed their specified values. They can then reduce usage or inform members of the group that are generating excessive charges.

To create a balance monitor, customers define:

- All account and service balances they want to include in their balance monitor. This group of accounts and services forms a *monitor group*.

  See "About Balance Monitoring Groups".
- When to be alerted that the balance total is too high or approaching the maximum.

  See "Alerting Customers When Monitored Balances Cross Limits or Thresholds".

About Balance Monitoring Groups

Customers specify which account balances or service balances they want to track by creating a balance monitoring group. Each group includes the following:

- **The owner**. Group owners have permission to view group balances, add or remove group members, and add or change threshold and credit limit values.

  Group owners can be accounts or services and can be anywhere in a hierarchy chain. For example, an owner could be a parent account in an account hierarchy or a paying or nonpaying child account.

- **All members**. Any of the following can be members of a monitor group:
  - Accounts. Members of a balance monitoring group can include parent accounts, paying child accounts, and nonpaying child accounts. BRM monitors the balances for all services under a member account. For example, if a member account contains a GSM service and a GPRS service, BRM adds the current balance for both services to the group balance.
  - Services. Members of a balance monitoring group can include services, such as the GSM service, for a group of accounts. When the member is in a service, BRM monitors balances for that service only. For example, a company that pays all GSM services for its employees could create a balance monitoring group that includes each employee's GSM service but excludes any other services owned by the employee accounts.

  Accounts and services can belong to one or more balance monitoring groups. BRM tracks the groups to which an account or service belongs and then accesses this information during the rating process to determine where to apply balance impacts.

- **The monitoring group type**. Monitoring group types specify the type of members that the group can contain. For example, group membership can be restricted to nonpaying child accounts and their services, or it can include both paying and nonpaying child accounts and their services. Table 23-1 shows the group types supported by balance monitoring.

Table 23-1 Supported Group Types

Group Type	Description
Hierarchy Credit Exposure (H_CE)	Supports *near* real-time monitoring of the group's balance. Members include paying and nonpaying child accounts and their services.
Payment Responsible Credit Exposure (PR_CE)	Supports *near* real-time monitoring of the group's balance. Members include nonpaying child accounts and their services.
Subscription Credit Exposure (SUB_CE)	Supports *near* real-time monitoring of the group's balance. Members include services in a subscription group.

Table 23-1 (Cont.) Supported Group Types

Group Type	Description
Payment Responsible Real-Time Credit Enforcement (PR_RTCE)	Supports real-time monitoring of the group's balance. Members include nonpaying child accounts and their services. Allows you to change the credit limit settings for the group's owner and members to do the following: – Roll up the credit limits from one or more child bill units in the group to the owner's bill unit. For example, assume a group's owner has a $1000 credit limit and the three child group members each have $100 credit limits. If you specify to roll up the credit limits of all three child group members, the group owner's credit limit would change to $1300. – Nullify a child's credit limit after it is rolled up to the owner. If you specified to nullify the credit limits in the previous example, BRM would change the credit limit for all three child group members to NULL. If you specified to maintain the credit limits, BRM would keep the credit limit for all three child group members at $100.

When a customer attempts to add a member to a balance monitoring group, BRM validates the member against rules you specify for each monitoring group type. The default implementation contains no validation rules for these monitoring group types, but you can create new group types or create your own rules by customizing the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode. For information, see "Validating the Members of a Balance Monitor Group" in *BRM Opcode Guide*.

About the Balances of a Monitor Group

The balance for a monitor group is a running total that includes all balance impacts generated by its members. BRM updates the balance when a group member generates a usage event or when you create or modify a monitor group.

BRM stores the total balance for a particular monitor group in **/balance_group/monitor** objects in the database.

Monitored balance totals include the following:

- **All standard Accounts Receivable (A/R) impacts**, such as these:
  - Discounts
  - Account-level adjustments and disputes
  - Payments
  - Refunds
  - Write-offs

  For more information, see "Managing Balances" in *BRM Managing Accounts Receivable*.
- **Taxes**. Monitored balance totals include taxes that are applied during the real-time rating process only. Any taxation that is deferred to the billing process is not included in the balance total.
- **Sponsored or charge sharing charges**. Monitored balance totals include charges that apply to the sponsor's account only.

  When an account belongs to a charge sharing or sponsor group, its balances may be paid by more than one account. For example, an employee may pay his own wireless usage fees but have his monthly subscription fees paid by his company. When this occurs, each

balance monitor tracks only the charges for which it is responsible. In this example, the company's balance monitor tracks the subscription fee, and the employee's balance monitor tracks all usage charges.

BRM updates the balance of a monitor group at the following times:

> **Note:**
>
> For PR_RTCE monitor types, you do not need to run **pin_monitor_balance**. This monitor group balance is updated in real time whenever an events occurs.

- **When a balance monitor is created**. When you first create a balance monitor and run the **pin_monitor_balance** utility, BRM automatically calculates the group balance. BRM retrieves the current balance of each member and adds it to the group balance.
- **When members are added to or removed from a monitor group**. When a member is added to a monitor group, BRM automatically adds the member's current balance to the group balance when **pin_monitor_balance** is run. Likewise, when a member is removed from a monitor group and **pin_monitor_balance** is run, BRM automatically decreases the group balance.
- **As a result cycle-fee rating**. When rating events, BRM calculates any monitor balance impacts immediately. However, the impacts are not added to the group balance until you run the balance monitor utility. Therefore, there is a small lag between the time an event occurs and the time it impacts the group balance.

Alerting Customers When Monitored Balances Cross Limits or Thresholds

To be alerted when the balance of their monitor group is approaching predefined values, customers must set up thresholds and credit limits. BRM then tracks the group balances and generates notification events when the balances cross above or below the threshold value or credit limit.

BRM checks whether a group balance has crossed a threshold or a credit limit whenever the following occur:

- The balance of a monitor group is updated
- Monitor group owners add or change limit and threshold values

About Setting Limits and Thresholds

To be alerted when a monitored balance reaches a certain value, customers must define the following for each balance monitor:

- **Credit floor**. The credit floor specifies the starting point for a threshold value. The default credit floor is NULL. You must set the credit floor to a non-NULL value.
- **Credit limit**. The credit limit specifies the ending point for a threshold value. BRM uses the credit limit to calculate the threshold value, which is a percentage of the credit limit. BRM generates a notification event when the group balance crosses above or below the credit limit.

> **Note:**
>
> *Balance monitor credit limits* are separate from *account credit limits*. Account credit limits specify the maximum balance for an individual account and affect the way BRM handles authorizations and rating. Monitor credit limits are used only to calculate threshold values and do not affect authorization and rating.

- **Thresholds**. Thresholds specify the balance totals that trigger an alert to CSRs and monitor group owners. Owners specify thresholds in the following ways:
  - As a percentage of the credit limit in 5% increments (75%, 80%, 85%, and so on). To be notified when the credit limit is reached, select the 100% threshold setting.
  - As a fixed value, such as $90 or 50 minutes.

For example, a customer might set a monitor group's credit floor to $0, credit limit to $100, and thresholds to 70%, 90%, and 100%. This causes BRM to generate notification events whenever the monitored balance crosses above or below $70, $90, and $100.

You must set a value for each parameter. BRM cannot send notification events when any of the following are true: the credit floor is NULL, the credit limit is infinite or undefined, or the threshold is undefined.

> **Note:**
>
> For payment responsibility real-time credit enforcement (PR_RTCE) balance monitoring groups, you can also specify the following:
>
> - Whether to roll up a member's account credit limit to the owner of the PR_RTCE balance monitoring group.
> - Whether to set a member's account credit limit to NULL after it is rolled up to the owner of the PR_RTCE balance monitoring group.

You define a balance monitor's credit floor, credit limit, and threshold values when you create the balance monitor in a custom client application. BRM then stores the values in a **/config/credit_profile** object in the BRM database. See "Managing Balance Monitors" in *BRM Opcode Guide* for more information.

About Notification Events for Balance Monitoring

BRM generates notification events whenever a monitored balance crosses over or under a credit limit or threshold. This enables you to make BRM perform different actions depending on the event type. For example, you can customize BRM to send an email to both you and the group owner when the group balance crosses above a threshold, but to send an email only to the group owner when the balance falls below a threshold.

About the Number of Notification Events

BRM generates one notification event for each monitored balance that crosses a threshold or credit limit. When one event causes a balance to cross multiple thresholds, BRM generates only one event that lists all settings that were breached.

For example, member A belongs to balance monitors 1 and 2. When member A generates a usage event that crosses the 50% and 55% thresholds for monitor 1, and the 40% threshold for monitor 2, BRM generates only two events:

- The first notification event specifies that the balance of monitor 1 surpassed both the 50% and 55% thresholds.
- The second notification event specifies that the balance of monitor 2 surpassed the 40% threshold.

Information Included in Balance Monitor Notification Events

Each balance monitor notification event includes the following information:

- **Target balance monitor**.
- **Resource ID**.
- **Alert type**: Credit limit or threshold breach.
- **Reason for the breach**:
  - **Upward breach**. The monitored balance crossed above a threshold value.
  - **Downward breach**. The monitored balance crossed below a threshold value, for example, when the customer makes a payment.
  - **Upward reset**. The owner of the balance monitor increased a threshold value, causing the balance to cross below a threshold. For example, this occurs when the current balance is $50 and you change the value of threshold 1 from $40 to $60. In this case, the $50 balance crosses below the new setting for threshold 1.
  - **Downward reset**. The owner of the balance monitor decreased a threshold value, causing the balance to pass above a threshold. For example, this occurs when the current balance is $40 and you change the value of threshold 1 from $50 to $25. In this case, the $40 balance passes above the new setting for threshold 1.
- **Monitor type**: Hierarchy credit exposure, paying responsibility credit exposure, or service level.
- **Source of the breach**.

> **Note:**
>
> BRM does not provide a source ID when the alert is due to a threshold reset or group member addition.

- **List of thresholds or credit limits that were breached**. For example, 50% and 70%. For credit limit breaches, this field is set to 100%.

Providing Your Customers with Access to Monitor Balances

Balance monitor owners can access the balance for their monitor groups in real-time by using a Web interface, such as a Web-enabled phone or computer. When monitor owners log in, BRM collects their account information and optionally a date range and then retrieves the following for each balance monitor:

- Beginning balance of the monitor group on the start date
- Ending balance of the monitor group on the end date

To provide your customers and CSRs with access to real-time monitor balances, you must customize your client interface to use the BRM API. For information, see "Displaying Balance Monitor Information in Client Applications" in *BRM Opcode Guide*.

About Creating and Maintaining Balance Monitors

You create and modify balance monitors by using a third-party customer relationship management (CRM) application. To implement balance monitoring in your CRM application, you must configure it to accept the following information for each balance monitor and pass it to the BRM API:

- Balance monitor name
- Balance monitor owner
- List of group members
- The monitor group type
- Credit limit (optional)
- Credit floor (optional)
- Threshold settings (optional)

The process that BRM uses to create or modify a balance monitor depends on whether you implement it in your system with or without Automated Monitor Setup (AMS).

- **With AMS**, you can automate many of the balance monitor processes. AMS automatically adds and removes members from the balance monitor for you.
- **Without AMS**, you add and remove members manually. You must use this method to create balance monitors for custom monitor types.

BRM creates balance monitors as follows:

1. Takes as input all information about the balance monitor, such as the monitor owner and threshold settings.
2. Creates a bucket to store the balance of the monitor group.
3. Adds members to the balance monitoring group. BRM determines whether the following is true:
   - AMS is enabled.
   - The monitor type is H_CE, PR_CE, SUB_CE, or PR_RTCE.

   If both are true, BRM uses AMS to add members. BRM uses the monitor owner and monitor type to find all appropriate child accounts and services, and adds them to the monitor. See "About Using AMS to Manage Balance Monitors Automatically".

   If either one is false, BRM adds members without AMS. BRM takes as input the list of members and validates them against custom criteria. All members that pass validation are added to the balance monitor. See "Managing Balance Monitors Without AMS".
4. Retrieves the current balance of all members and adds it to the group balance.

About Using AMS to Manage Balance Monitors Automatically

You configure BRM to automatically add and remove members from balance monitors, based on the monitor group type, by using AMS. AMS is a group of opcodes that automate many of the balance monitor processes, making balance monitoring easier to set up and use.

AMS creates and updates balance monitors by following the organizational hierarchy of accounts and services. This synchronizes the membership between a hierarchy and a monitor group.

> **Note:**
>
> AMS works only for the default monitor group types: H_CE, PR_CE, SUB_CE, and PR_RTCE. For custom group types, you must create balance monitors without AMS. See "Managing Balance Monitors Without AMS".

AMS automatically adds and removes members from balance monitors when the following occur:

- **A balance monitor is created**. AMS takes as input the parent (owner) account or service and the monitor type, and then searches through the parent's lineage to find accounts and services that match the criteria for the monitor type. All subordinate accounts and services that meet the criteria are added as members.

  For example, when users create a paying responsibility-type monitor, AMS adds to the monitor the parent account and its services and all subordinate nonpaying child accounts and their services.
- **An account hierarchy or subscription group is changed**. AMS automatically adds or removes members from any associated balance monitors, based on the monitor type. For example, when you add a nonpaying child account to a hierarchy, AMS automatically:
  - Finds all parent accounts at a higher level in the hierarchy than the specified child account.
  - Finds all hierarchy and paying responsibility balance monitors associated with each parent account.
  - Adds the child account and its services to each balance monitor.

When a balance monitor is created or a hierarchy changes, the following occurs:

1. BRM generates a notification event.
2. The BRM event notification system calls one of the AMS opcodes.
3. The AMS opcodes add or remove members from the appropriate monitor group and update the monitored balance.

Managing Balance Monitors Without AMS

When creating or updating balance monitors without AMS, users are responsible for manually adding and removing members. BRM does not verify that balance monitors include all appropriate child accounts and services in a hierarchy or subscription group nor does it check whether a hierarchy or subscription group changes over time.

Users must add or remove members when the following occur:

- **A balance monitor is created**. Users specify the individual accounts and services to add to the balance monitor.
- **An account hierarchy or subscription group is changed**. Users must manually update any balance monitors associated with the hierarchy or subscription group. When an account is added to a hierarchy, users must manually add the account to all balance

monitors associated with the group. Likewise, when an account is removed from a hierarchy, users must manually remove the account from any associated monitors.

To manage balance monitors without AMS, see "Creating and Maintaining Balance Monitors Manually".

About Processing Balance Impacts

BRM processes balance impacts for balance monitoring as follows:

1. BRM rates events and collects any monitored balance impact data.

   See "About Balance Monitoring and Real-Time Rating"

2. BRM publishes monitored balance impacts to the monitor queue.

   See "Understanding the Monitor Queue".

3. The **pin_monitor_balance** utility retrieves and processes data from the monitor queue. If a balance crosses a credit limit or threshold, the utility generates a threshold breach notification event.

   See "Using pin_monitor_balance to Update Monitored Balances".

4. If a threshold breach notification event is generated, the BRM event notification system calls the specified opcode, which processes the event data and passes information to your custom client application.

   See "About Using Event Notification to Alert Customers".

About Balance Monitoring and Real-Time Rating

Real-time rating processes monitored balances as follows:

1. Rates the event and applies any discounts.
2. Determines whether the event owner belongs to any monitor groups.
3. Determines whether balance monitoring is enabled.
4. Generates monitor balance impacts for each monitor group listed in the **/ordered_balgrp** object. The monitor balance impact includes the following data:
   - Event owner
   - Amount
   - Resource ID
   - Balance monitor (POID of the **/balance_group/monitor** object)

Understanding the Monitor Queue

> **Note:**
>
> The monitor queue is not applicable to PR_RTCE monitor types, because their balance impacts happen in real time.

The monitor queue (**/monitor_queue**) holds all monitor balance impacts generated by rating. The **pin_monitor_balance** utility retrieves this information and updates the balances of the associated monitor groups.

For performance reasons, the **pin_monitor_balance** utility does not delete **/monitor_queue** objects after they are retrieved from the queue. Instead, the utility flags the **/monitor_queue** objects as "processed" to prevent the objects from being consumed again.

BRM tracks the status of **/monitor_queue** objects by using its PIN_FLD_STATUS field:

- When the field is set to **0**, the object has not been processed by the **pin_monitor_balance** utility.
- When the field is set to a **nonzero value**, the object has been processed by the utility.

Using pin_monitor_balance to Update Monitored Balances

> **Note:**
>
> The **pin_monitor_balance** utility is not applicable for PR_RTCE monitor types, because their balance impacts happen in real time.

You use the **pin_monitor_balance** utility to apply balance impacts to a balance monitor. This utility is a multithreaded application (MTA) that retrieves a list of all **/monitor_queue** objects in the BRM database and then spawns child threads to process each object individually.

You can configure the utility to run with one thread or multiple threads:

- One thread enables the utility to process events in the order that they occurred but decreases processing performance.
- Multiple threads increase performance but may cause the utility to process events out of order. The default is five threads.

To configure the number of threads, see "Configuring pin_monitor_balance to Process Events in the Order Created".

You can run **pin_monitor_balance** manually or configure a scheduler, such as **cron**, to run it at predefined intervals. For more information, see "Running pin_monitor_balance to Update Monitored Balances".

The **pin_monitor_balance** utility performs the following tasks:

1. Retrieves all unprocessed **/monitor_queue** objects from the queue (that is, all objects with the PIN_FLD_STATUS field set to **0**).
2. Retrieves the **/event** object associated with the **/monitor_queue** object to obtain the following:
   - Monitor balance impacts
   - Sub-balance impacts
   - Target balance monitors (**/balance_group/monitor** objects)
   - Event owner
3. Calls the PCM_OP_BAL_APPLY_MONITOR_IMPACTS opcode, which performs the following:

a. Updates the monitored balance (**/balance_group/monitor**).

b. Determines whether any balance monitor thresholds have been breached by comparing the monitored balance to the threshold and credit limit values stored in the **/config/credit_profile** object. If any thresholds or limits were breached, the opcode generates one of these notification events:

   - **/event/notification/threshold**: Indicates that the balance crossed *above* a threshold value or credit limit.
   - **/event/notification/threshold_below**: Indicates that the balance crossed *below* a threshold value or credit limit.

c. Sets the **/monitor_queue** object's PIN_FLD_STATUS field to a nonzero value to prevent it from being processed again by the utility.

For more information about the opcode, see "Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications" in *BRM Opcode Guide*.

About Using Event Notification to Alert Customers

BRM uses event notification to alert monitor owners when balances cross a threshold or credit limit. After an **/event/notification/threshold** or **/event/notification/threshold_below** event occurs, the BRM event notification system does the following:

1. Calls the opcode associated with the event in your system's event notification list. By default, the event notification system calls the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode.

   See "About the Event Notification List" in *BRM Developer's Guide*.

2. The opcode processes the event data and passes information to your custom client application.

To enable this, you must configure the BRM event notification system, the opcode (if it is a policy opcode), and your custom client application to process these events and alert balance monitor owners. See "Configuring Event Notification for Balance Monitoring".

Configuring Your BRM System for Balance Monitoring

To set up your BRM system for balance monitoring, do the following:

1. Enable balance monitoring in BRM.

   See "Enabling Balance Monitoring in BRM".

2. Enable AMS in BRM.

   See "Enabling AMS in BRM".

3. Configure BRM to perform appropriate follow-up operations when credit limits and thresholds are crossed.

   See "Configuring Event Notification for Balance Monitoring".

4. Configure **pin_rerate** to wait a specified amount of time before starting the rerating process.

   See "Specifying a Wait Time Before Rerating Events".

5. For H_CE, PR_CE, and SUB_CE monitoring group types only, do the following:

   - (Optional) Configure **pin_monitor_balance** to process monitor balance impacts in the order that they were created.

See "Configuring pin_monitor_balance to Process Events in the Order Created".

- Run **pin_monitor_balance** on a regular basis.

  See "Running pin_monitor_balance to Update Monitored Balances".

6. Configure your custom client application to call the appropriate opcodes for managing balance monitoring.

   See "Implementing Balance Monitoring in Custom Client Applications".

7. For payment responsible real-time credit enforcement (PR_RTCE) balance monitoring groups only, configure the following:

   - For each child group member, whether to roll up its credit limit to the group's owner.

     See "Rolling Up Child Credit Limits to Owners in PR_RTCE Groups".

   - Whether to nullify the credit limits of child group members after their credits are rolled up to the group's owner.

     See "Nullifying Credit Limits for PR_RTCE Child Members".

After your system is set up, you can use a custom client application to create balance monitors and balance monitoring groups for your customers.

Enabling Balance Monitoring in BRM

By default, balance monitoring is disabled in BRM. You can enable this feature by modifying a field in the **multi-bal** instance of the **/config/business_params** object. When balance monitoring is enabled, BRM monitors currency resources.

You modify the **/config/business_params** object by using the **pin_bus_params** utility (see "pin_bus_params" in *BRM Developer's Guide*).

To enable balance monitoring:

1. Use the following command to create an editable XML file from the **multi-bal** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

   This command creates the XML file named **bus_params_multi_bal.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **BalanceMonitoring** to **enabled**:

   ```
   <BalanceMonitoring>enabled</BalanceMonitoring>
   ```

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing **multi-bal** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM balance monitoring configuration.

3. Save the file and change the file name from **bus_params_multi_bal.xml.out** to **bus_params_multi_bal.xml**.

4. Use the following command to load this change into the **/config/business_params** object:

```
pin_bus_params bus_params_multi_bal.xml
```

You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

6. Stop and restart the Connection Manager (CM).

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Enabling AMS in BRM

By default, AMS is disabled in BRM. You can enable this feature to automate many of the balance monitoring processes by modifying a field in the **subscription** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility (see "pin_bus_params" in *BRM Developer's Guide*).

To enable AMS:

1. Use the following command to create an editable XML file from the **subscription** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSubscription bus_params_subscription.xml
   ```

   This command creates the XML file named **bus_params_subscription.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **AutomatedMonitorSetup** to **enabled**:

   ```
   <AutomatedMonitorSetup>enabled</AutomatedMonitorSetup>
   ```

   > **Caution:**
   >
   > BRM uses the XML in this file to overwrite the existing **subscription** instance of the **/config/business_params** object. If you delete or modify any other parameters in the file, these changes affect the associated aspects of the BRM balance monitoring configuration.

3. Save the file and change the file name from **bus_params_subscription.xml.out** to **bus_params_subscription.xml**.

4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

6. Stop and restart the Connection Manager (CM).

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Configuring Event Notification for Balance Monitoring

When the balance of a monitoring group crosses a threshold or a credit limit, BRM generates the following notification events. You must configure the BRM event notification feature to call opcodes that perform the appropriate follow-up operations when these events are generated:

- **/event/notification/threshold**: This event indicates that the group's balance reached or exceeded a threshold value; for example, when a purchase causes the balance to go over a 70% threshold.
- **/event/notification/threshold_below**: This event indicates that the group's balance passed below a threshold value; for example, when the customer pays a bill. This event is generated only when the group's balance is less than the threshold value.

By default, when these events occur, the event notification system calls the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode (opcode number **301**). The policy opcode then sends a message to your customers using the Email Data Manager (DM). However, you can customize this policy opcode to perform other actions.

> **Note:**
>
> To find an opcode's number, see the opcode header files in *BRM_home***/include/ops**.

To configure the event notification feature for balance monitoring:

1. Open your *BRM_home***/sys/data/config/pin_notify** file.
2. Add the following lines:

```
301     0       /event/notification/threshold
301     0       /event/notification/threshold_below
```

> **Note:**
>
> If you want the events to trigger calls to other opcodes, change the opcode number. The number of each balance monitoring opcode is in the balance monitor opcode header file (*BRM_home***/include/ops/monitor.h**).

3. If you enabled AMS, ensure that the following lines in bold are uncommented:

```
# Automated Monitor Setup related event notification
# Uncomment these lines when AMS feature is enabled
7853    0       /event/group/sharing/monitor/create
7854    0       /event/notification/service/pre_purchase
```

```
7855    0       /event/notification/service/pre_purchase
7856    0       /event/notification/service/pre_purchase
7857    0       /event/customer/billinfo/modify
7855    0       /event/customer/billinfo/modify
7857    0       /event/group/member
7854    0       /event/group/member
7857    0       /event/notification/bal_grp/modify
7855    0       /event/notification/bal_grp/modify
7854    0       /event/audit/subscription/transfer
3787    0       /event/billing/monitor/update
```

> **Note:**
>
> If the event is configured to trigger opcodes **7853** and **1301** (PCM_OP_PUBLISH_GEN_PAYLOAD), list the events in this order:
>
> ```
> 1301    /event/group/sharing/monitor/create
> 7853    /event/group/sharing/monitor/create
> ```

4. Save and close your **pin_notify** file.
5. If your system has multiple configuration files for event notification, merge them.

   See "Merging Event Notification Lists" in *BRM Developer's Guide*.
6. Load your final event notification list into the BRM database:
   a. Go to the directory that contains your final **pin_notify** file.
   b. Run the following command to run the **load_pin_notify** utility:

      load_pin_notify *configFileName*

      where *configFileName* is the name of your final **pin_notify** file. If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file.
   c. Stop and restart the Connection Manager (CM).

Specifying a Wait Time Before Rerating Events

You must configure **pin_rerate** to wait a specified amount of time before rerating events.

To configure a wait time:

1. Open the **pin_rerate** configuration file (*BRM_home*/**apps/pin_rerate/pin.conf**) in a text editor.
2. Add the following entry and specify the delay time in seconds.

   ```
   - pin_rerate        delay        300
   ```

   The default is **0**.
3. Save and close the file.

Configuring pin_monitor_balance to Process Events in the Order Created

> **Note:**
>
> This step is not required for PR_RTCE monitor group types.

To process monitor balance impacts in chronological order, configure the **pin_monitor_balance** utility to run with only one thread. For more information, see "Using pin_monitor_balance to Update Monitored Balances".

To process monitor balance impacts in chronological order, perform the following tasks:

1. Open the **pin_monitor_balance** configuration file (*BRM_home***/apps/pin_monitor/pin.conf**) in a text editor.
2. Change the following entry to **1**.

```
- pin_mta   children   1
```

3. Save and close the file.

Running pin_monitor_balance to Update Monitored Balances

> **Note:**
>
> This step is not required for PR_RTCE monitor group types.

To update monitored balances, perform the following:

1. Go to the *BRM_home***/apps/pin_monitor** directory.
2. Run the **pin_monitor_balance** utility:

> **Note:**
>
> To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

```
pin_monitor_balance [-d] [-v]
```

For more information, see "pin_monitor_balance".

Implementing Balance Monitoring in Custom Client Applications

To monitor balances, your Customer Service Representative (CSR) must be able to perform the following tasks:

- Specify the group of account and service balances to monitor

- Update the group by adding or deleting members
- View the balance monitors owned by an account
- View the current balance of a balance monitoring group
- Receive alerts when a balance monitoring group's credit limit or threshold is crossed

To implement balance monitoring in your client application, add the following functionality to your application by using BRM opcodes:

1. Create or update balance monitors and balance monitoring groups.
   - To have BRM create them automatically for you, see "Creating and Maintaining Balance Monitors Automatically".
   - To create them manually, see "Creating and Maintaining Balance Monitors Manually".
2. Display the balances for a balance monitoring group or a list of monitors owned by an account or service.

   See "Displaying Balance Monitor Information in Client Applications" in *BRM Opcode Guide*.
3. Send notifications when a group's credit limit or threshold is crossed.

   See "Updating Monitor Balances and Sending Credit Limit/Threshold Breach Notifications" in *BRM Opcode Guide.*
4. (Optional) Specify whether to include item transfers in the balances of monitor groups.

   See "Specifying Whether Item Transfers Affect Balance Monitors".

Your user interface must be designed to collect the information needed for creating or updating the relevant objects and pass the appropriate fields in the input flist to the opcodes.

Creating and Maintaining Balance Monitors Automatically

You can configure a custom client application to automatically manage balance monitors by calling BRM opcodes. To do so, add the following functionality to your custom client application:

1. Manage balance monitors.

   To create and modify balance monitors (**/balance_group/monitor** objects), call PCM_OP_CUST_SET_BAL_GRP. See "Creating and Updating Balance Monitors" in *BRM Opcode Guide*.

   > **Note:**
   >
   > You cannot delete **/balance_group/monitor** objects. However, you can deactivate the monitor group by deleting its associated **/group/sharing/monitor** object. See "Deactivating Balance Monitors" in *BRM Opcode Guide*.

2. Manage balance monitoring groups:
   - To create balance monitoring groups, call PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE.

     When the balance monitor group is created successfully, the opcode generates an **/event/group/sharing/monitor/create** event.

- To modify balance monitoring groups, call PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY.

  When the balance monitor group is modified successfully, the opcode generates an **/event/group/sharing/monitor/modify** event.

- To delete balance monitoring groups, call PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE.

  When the balance monitor group is deleted successfully, the opcode generates an **/event/group/sharing/monitor/delete** event.

The generated events trigger the BRM event notification system to call the AMS opcodes, which automatically add and remove members from balance monitoring groups (**/group/sharing/monitor** objects) and update each member's **/ordered_balgrp** object.

For more information about these opcodes, see "Managing Sharing Groups" in *BRM Opcode Guide*.

Creating and Maintaining Balance Monitors Manually

To create and maintain balance monitors manually (without AMS), you must configure your custom client application to call the following opcodes:

1. PCM_OP_CUST_SET_BAL_GRP: This opcode creates and updates balance monitors (**/balance_group/monitor** objects).

   See "Managing Balance Monitors" in *BRM Opcode Guide*.

2. PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE: This opcode creates balance monitoring groups (**/group/sharing/monitor** objects).

   You must pass all group members in the input flist.

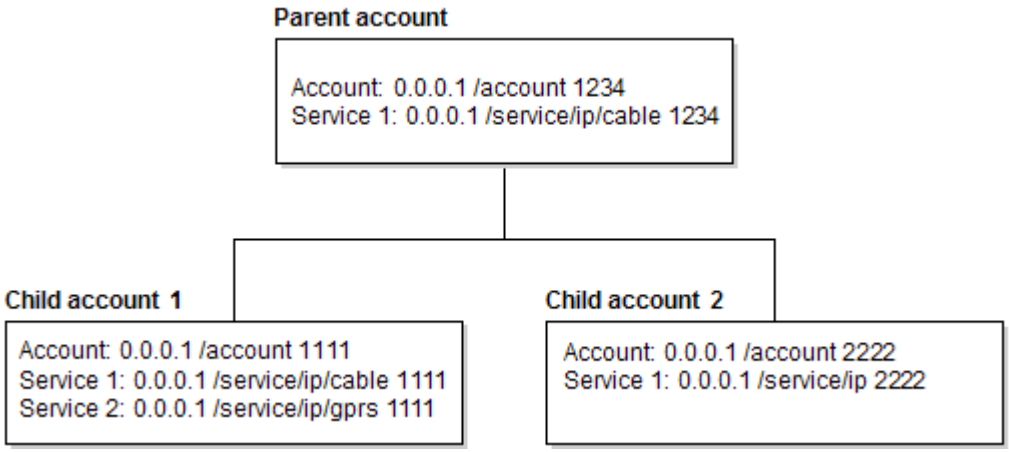
   - To create, modify, or delete a balance monitoring group, see "Managing Sharing Groups" in *BRM Opcode Guide*.
   - To validate potential members before adding them to a balance monitoring group, see "Validating the Members of a Balance Monitor Group" in *BRM Opcode Guide*.

3. PCM_OP_SUBSCRIPTION_ORDERED_BALGRP: This opcode creates and updates each member's list of sharing groups to which they belong (in **/ordered_balgrp** objects).

   You must create or update the **/ordered_balgrp** object associated with each account and service that you want to include in a balance monitoring group. See "Managing Ordered Balance Groups" in *BRM Opcode Guide*.

   For example, assume a balance monitor includes the following account hierarchy shown in Figure 23-1.

Figure 23-1 Example Account Hierarchy in a Balance Monitor



To include in the balance monitor the balance of all three accounts and their services, create or update the **/ordered_balgrp** object associated with each of the following:

- **/account 1234**
- **/service/ip/cable 1234**
- **/account 1111**
- **/service/ip/cable 1111**
- **/service/ip/gprs 1111**
- **/account 2222**
- **/service/ip 2222**

Specifying Whether Item Transfers Affect Balance Monitors

BRM updates balance monitor totals when you perform item adjustments, settlements, and disputes. By default, BRM also updates balance monitor totals when you perform item transfers. This can cause BRM to apply double the dispute, adjustment, or settlement amount to a balance monitor total when correcting misapplied charges. For example, when a customer disputes a misapplied charge, BRM reduces the balance monitor total when the charge is disputed and then again when the charge is transferred from the customer's account to another account.

You can configure BRM to not apply balance impacts from item transfers to balance monitor totals by passing the optional PIN_FLD_APPLY_MONITOR input flist field to the PCM_OP_BILL_ITEM_TRANSFER opcode:

- When PIN_FLD_APPLY_MONITOR is set to **0**, balance impacts from item transfers *are not* added to balance monitor totals.
- When PIN_FLD_APPLY_MONITOR is set to **1** or not passed, balance impacts from item transfers *are* added to balance monitor totals. This is the default.

For more information about adjustments, disputes, and settlements, see "Making Adjustments" in *BRM Managing Accounts Receivable*.

Rolling Up Child Credit Limits to Owners in PR_RTCE Groups

PR_RTCE balance monitoring groups allow you to roll up the credit limits from one or more child bill units in the group to the owner's bill unit.

To configure BRM to roll up the credit limit from a child's bill unit to a parent's bill unit in a balance monitoring group:

- **Using Billing Care**: Use the **Roll Up Credit Limit** check box when creating a child account, modifying a child account's credit limit, adding a nonpaying child to a billing hierarchy, or adding a paying parent to a billing hierarchy.

  See "Working with Credit Limits, Credit Floors, and Thresholds" and "Billing Hierarchies" in *Billing Care Online Help*.

- **Using a custom client application calling opcodes**: Setting PIN_FLD_TYPE_STR set to **PR_RTCE** and PIN_FLD_ROLL_UP_CREDIT_LIMIT set to **1** when creating a balance monitoring group, creating a child member's account, or modifying a child member's account.

  See "Creating a Balance Monitor Group" and "How BRM Handles Consumption Rules and Credit Limits" in *BRM Opcode Guide*.

- **Using a custom client application calling the Billing Care REST API**: Setting the **rollUp** field in the request payload of the following operations:
  - **Calling the Add a Bill Unit to a Hierarchy**
  - **Create Credit Limits for a Balance Group**
  - **Create a Bill Unit**
  - **Update a Bill Unit for an Account**

  See *REST API Reference for Billing Care*.

Nullifying Credit Limits for PR_RTCE Child Members

For PR_RTCE balance monitoring group types, you can configure whether each child member's credit limit is set to NULL or remains unchanged after their credit limit is rolled up to the group's owner. By default, their credit limit is nullified.

You configure this functionality by modifying the **rating** instance of the **/config/business_params** object using the **pin_bus_params** utility (see "pin_bus_params" in *BRM Developer's Guide*).

To configure whether to nullify child member credit limits in PR_RTCE groups, do the following:

1. Use the following command to create an editable XML file from the **rating** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

   This command creates the XML file named **bus_params_rating.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **ResetMemberCreditLimit** to the appropriate value:

   ```
   <ResetMemberCreditLimit>value</ResetMemberCreditLimit>
   ```

   where *value* is:

- **enabled** specifies that after a member's credit limit is rolled up to the group's owner, to set the member's credit limit to NULL. This is the default value.
- **disabled** specifies to not change the member's credit limit after it is rolled up to the group's owner.

3. Save the file and change the file name from **bus_params_rating.xml.out** to **bus_params_rating.xml**.
4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_rating.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.
6. Stop and restart the Connection Manager (CM).

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

24

Creating and Managing Profile Sharing Groups

Learn how to use profile sharing groups to enable an account or an account's service to share a profile with other accounts or services in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

To set up profile sharing groups, you should be familiar with extended rating attributes (ERAs). See "About Extended Rating Attributes" in *BRM Configuring and Running Billing*.

About Profile Sharing Groups

A profile sharing group enables an account or an account's service to share a profile with other accounts or services. A profile stores ERAs or other types of information about an account.

The group owner can be an account or a service. If an account is the owner, profiles from all the services owned by the account are available for sharing. If a specific service is the owner, only profiles of that service are available for sharing.

> **Note:**
>
> Only service profiles can be shared. Account profiles cannot be shared, even if the profile sharing group is owned by an account.

You can use profile sharing to share an ERA, such as a friends and family list, with multiple accounts or services. For example, a friends and family list can be set up for a GSM service owned by one account and then shared so that a GSM service belonging to other accounts can use the same list.

You can also use profile sharing groups to share custom profiles you create.

You set up profile sharing groups in one of these ways:

- Using Billing Care
- Using Customer Center

- Using a customized third-party client application calling BRM opcodes or the Billing Care REST API

For more information, see "Creating Profile Sharing Groups".

> **Note:**
>
> Profile sharing groups work similarly to charge and discount sharing groups, but profile sharing groups do not share balances and are not prioritized.

About Profile Sharing Group Membership

Only service profiles can be shared, but a profile sharing group's members can be accounts or services, as follows:

- **Account**: If an account is a member, any eligible event generated by the account can use a shared profile.
- **Service type**: When you specify a service type (for example, GSM) as a member, the events generated by all service instances within that service type are considered for shared profiles.

  You can also specify that a member account that has not yet purchased a service of that type is automatically eligible for participation in profile sharing if it buys the service in the future.

  > **Note:**
  >
  > The subtypes of the service type you specify do not become members. For example, if you specify the GSM service type as a member, specific GSM service instances such as GSM fax, GSM telephony, and so on do not.

- **Service instance**: When you specify a service instance, such as a specific phone, as a member, only the events generated by that instance are eligible for the shared profiles. Specify membership using a specific service instance if you want to exclude other services of that type from profile sharing.

  For example, if an account includes both work and personal phone services, a friends and family ERA might apply only to the personal phone.

  Use the PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_SERVICE opcode to customize adding a service to a sharing group. See "Adding Members to a Profile Sharing Group" in *BRM Opcode Guide*.

When setting up profile sharing groups:

- Profile sharing group members must be equivalent to the owner. If the owner is a service, then the members must be the same service type. If the owner is an account, the members must also be accounts.
- Members can have different charges or discount offers than the owners. To use of a shared profile, members need a charge or discount offer configured to use an ERA type or ERA label in the shared profile to give special charge pricing or discounts.
- When the status of a profile sharing group's owner account is changed to *inactive* or *closed*, the shared profiles are no longer available to group members. During rating, the

shared profiles are not retrieved. If an inactive owner account is activated, the shared profiles are once again available during rating.

- By default, all members of a profile sharing group must reside in the same database schema. To enable profile sharing groups to include members from multiple schemas, see "Enabling Sharing Groups to Support Multiple Schemas".
- By default, BRM does not validate profile sharing group members. You can customize the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode to add validation for profile sharing groups. See "Validating Profile Sharing Group Members " in *BRM Opcode Guide*.

Creating Profile Sharing Groups

To create a profile sharing group, you specify a group owner account or service, member accounts or services, and the list of the group owner's profiles that the members will share.

You create profile sharing groups as follows:

- Using Billing Care
- Using Customer Center
- Using a third-party client application calling BRM opcodes or the Billing Care REST API

The following guidelines apply when creating a profile sharing group:

- By default, all members of a profile sharing group must reside in the same database schema. To enable profile sharing groups to include members from multiple schemas, see "Enabling Sharing Groups to Support Multiple Schemas".
- The owner cannot be a member of its own group.
- The owner of a profile sharing group cannot also belong to a member-owned group. For example, if Account1 and Account2 both own profile sharing groups, and Account2 is a member of Account1's group, then Account1 cannot belong to Account2's group.
- If you add a service type as a member rather than a specific service instance, all instances of that service type become members. However, the subtypes of the service type do not become members.

  For example, if you specify the GSM service type as a member, all instances of GSM become members, but GSM fax, GSM telephony, and so on do not.

Adding a Profile Group to a Member's Ordered Balance Group

The **/ordered_balgrp** object stores the list of sharing groups to which an account or service belongs. This object is used with all types of sharing groups, but its significance varies:

- For charge and discount sharing groups, **/ordered_balgrp** controls the order in which events impact the group's balances. See "How Charges and Discounts Are Applied".
- For profile sharing groups, the order is not significant, so they are added to the end of the list in the PIN_FLD_ORDERED_BALGROUPS array.

When a profile sharing group is created or modified, the PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS policy opcode automatically adds the group to the **/ordered_balgrp** object of each account or service that is a member.

Modifying Profile Sharing Groups

You can modify profile sharing groups using Billing Care or Customer Center, or by customizing a third-party application. You can change a profile sharing group in the following ways:

- **Add members to the group**: When a member is added, shared profiles become available for events generated by that member.
- **Add shared profiles**: When profiles are added to a profile sharing group, the profiles are automatically available to group members. The group owner must own the added profiles and have valid dates.
- **Delete members from the group**: When a member is deleted, the group owner's profiles are no longer considered in rating or discounting the former member's events.
- **Delete shared profiles**: A deleted shared profile is removed from the group's profiles list and is no longer available to members.
- **Change the owner of the profile sharing group**: When the owner changes, members use the new owner's shared profiles instead of the old owner's shared profiles.

Deleting Profile Sharing Groups

You can delete a profile sharing group when the group owner's account is closed, or a CSR deletes the sharing group. When a profile sharing group is deleted, members can no longer use the profiles that the owner had shared.

When a profile sharing group is deleted, BRM also deletes the profile sharing group from each member's ordered balance group. For more information about ordered balance groups, see "Adding a Profile Group to a Member's Ordered Balance Group".

You delete profile sharing groups as follows:

- Using Billing Care
- Using Customer Center
- Using a third-party client application calling BRM opcodes or the Billing Care REST API

Enabling Sharing Groups to Support Multiple Schemas

By default, all members of a sharing group must reside in the same database schema. If you want to include a member from a different schema, you must first use **pin_amt** to migrate the member to the same schema as all other sharing group members.

Alternatively, you can enable sharing groups to include members from other database schemas. To do so, use the **pin_bus_params** utility to modify the **CrossSchemaSharingGroup** business parameter in the **system** instance of the **/config/business_params** object.

To enable group members to reside in multiple schemas:

1. Use the following command to create an editable XML file from the **system** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsSystem bus_params_system.xml
   ```

This command creates the XML file named **bus_params_system.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Set **CrossSchemaSharingGroup** to **enabled**:

   ```
   <CrossSchemaSharingGroup>enabled</CrossSchemaSharingGroup>
   ```

3. Save the file and change its name to **bus_params_system.xml**.

4. Use the following command to load this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_system.xml
   ```

   You should run this command from the *BRM_home*/**sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

5. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

6. Stop and restart the CM.

   For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

25

Creating and Managing Service Groups

Learn how to use service groups to enable customers to purchase, transfer, inactivate, and cancel services as a group; and to share charges, discounts, and promotions among a group of subscribers in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Grouping Services by Subscription

In BRM, you can combine services into *service groups*, which are sets of services associated with a subscription, such as telephony and text messaging services associated with a wireless connection.

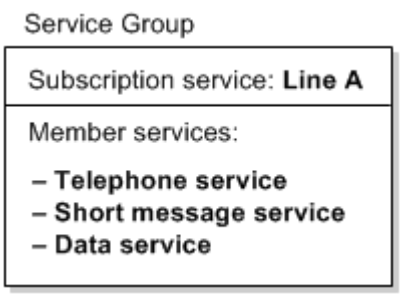
A service group consists of a *subscription service* and any number of *member services*. The subscription service can be a service that subscribers use, such as telephony or broadband access, or it can be a representational service with no associated usage fees.

When services are grouped by subscription, you can do the following:

- Enable customers to purchase, transfer, inactivate, and cancel services as a group.
- Apply discounts and promotions owned by the subscription service to the member services.
- Enable member services to benefit from the subscription service's membership in sharing groups.
- Associate member services with the devices owned by the subscription service.
- Keep separate balances for the service group.
- Create a separate bill and invoice for the subscription.

For example, a service group might consist of the services shown in Figure 25-1.

Figure 25-1 Example Service Group



About Service Groups

In service groups, subscription services and member services as follows:

- Sharing Charge and Discount Offers
- Sharing Balance Groups

Sharing Charge and Discount Offers

Subscription and member services share charge offers as follows:

- If the subscription service owns a charge offer but a member service does not, the member service uses the subscription service's charge offer to rate its balance impacts.
- If both the subscription service and a member service own charge offers, the member service uses its own charge offer. It cannot use the subscription service's charge offer.

Subscription and member services share discount offers as follows:

- If the subscription service owns a discount offer but a member service does not, the member service uses the subscription service's discount offer to rate its balance impacts.
- If both the subscription service and a member service own discount offers, the member service uses both its discount offer and the subscription service's discount offer.

Sharing Balance Groups

A balance group is a collection of sub-balances that track what customers owe or are credited with for one or more services. Services can use the default account balance group, or they can use a service balance group. See *BRM Managing Accounts Receivable* for more information.

If a subscription service has its own balance group:

- All member services that do not have a balance group use the subscription service's balance group.
- Any member service that has a balance group uses its own balance group instead of the subscription service's balance group.

About Using Charge and Discount Offers to Rate Service Group Usage

Both subscription services and member services can own charge and discount offers:

- When the subscription service owns charge and discount offers, the offers are used to rate member service usage.
- When the subscription service and a member service own charge offers, the charge offers' priority determines whether the subscription or member charge offer is used for rating.
- When the subscription service and a member service own discount offers, the discount offers' priority determines the order in which the discounts are applied.
- If a member service owns a discount offer but does not have its own balance group, its discount is applied to the subscription service's balance group.

You specify the priority of charge and discount offers when you create them in PDC or Pricing Center.

About Distributing Discounts to Member Services

Discounts owned by a subscription service can be distributed to member services as follows:

- **Through a discount sharing group:** When the subscription service and a member service have separate balance groups, the subscription service can share its discounts with the member service through a discount sharing group. The discount balance impact can be applied to a balance in the subscription service's balance group or in the member service's balance group. See "About Sharing Charges and Discounts in Service Groups".
- **Through inheritance:** When a member service shares the subscription service's balance group, the member service automatically inherits any discount purchased by the subscription service. The discount balance impact is applied to the subscription service's shared balance group.

> **Note:**
>
> When a billing-time discount grants a percentage off subscription fees based on the total usage of all services in the group, the total discount granted can be greater than intended. If you do not want billing-time discounts to be inherited, do one of the following:
>
> – Purchase them for member services instead of for the subscription service.
>
> – Configure BRM to disallow inheritance for billing-time discounts. See "Preventing Member Services from Inheriting Billing-Time Discounts".

Preventing Member Services from Inheriting Billing-Time Discounts

By default, billing-time discounts owned by a subscription service in a service group are automatically inherited by all member services that share the subscription service's balance group. When the discount is applied at billing time, it is applied individually to each member service.

If a subscription service owns a billing-time discount that grants a percentage off subscription fees, the total discount can be greater than intended when the discount is inherited. For example, a service group includes a subscription service and ten member services that inherit a billing-time discount. The discount provides .05% off the group's aggregated charges. If the aggregated charges total $1000 when billing is run, the discount grants $5 off to each member service that inherits the discount, and $5 off to the subscription service. The total discount is $55, which is an actual discount of .055% off the aggregated charges.

To prevent member services from inheriting billing-time discounts:

1. Open the CM **pin.conf** file in *BRM_home***/sys/cm**, where *BRM_home* is the directory where you installed BRM components.
2. Search for the following line:

   ```
   - fm_subscription btd_inheritance 1
   ```

3. Set the value of the **btd_inheritance** entry to **0**.

   The default is **1**, which means billing-time discounts are inherited.
4. Stop and restart the CM.

About Sharing Charges and Discounts in Service Groups

Charge and discount sharing groups are groups of services that share balances. They consist of a group owner and group members.

To set up charge or discount sharing for a service group (for example, to distribute 300 minutes to all services on a phone line), make the subscription service the sharing group owner, and make the member services the sharing group members.

The service type of the subscription service must match the service type to which the shared discount applies.

> **Note:**
>
> All services in a sharing group must have their own balance groups so BRM can track the balances they use and share.

See "About Charge and Discount Sharing Groups".

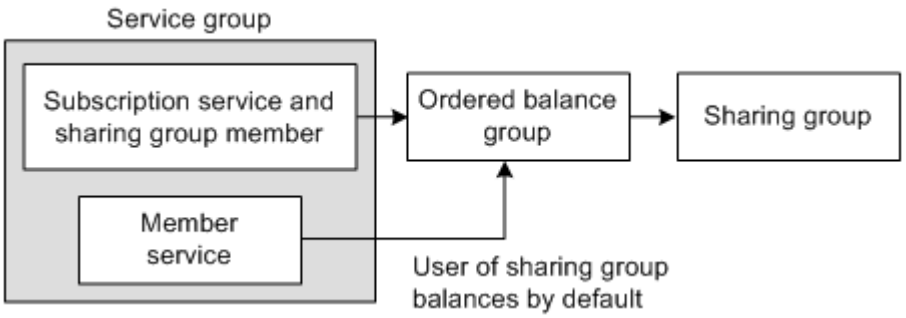
About Distributing Sharing Group Balances to Member Services

Each charge and discount sharing group member has an *ordered balance group*. See "About Ordered Balance Groups".

When a subscription service is a *member* in a sharing group, its member services may or may not share its ordered balance group:

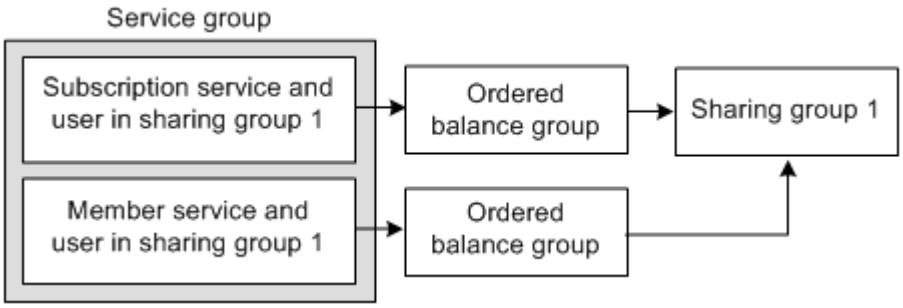
- A member service that is not a member of a sharing group uses the ordered balance group of its subscription service. By default, the member service inherits member status in the sharing group from the subscription service, and the member service has access to the sharing group's balances as shown in Figure 25-2.

Figure 25-2 Member Service Using Its Subscription Service's Ordered Balance Group



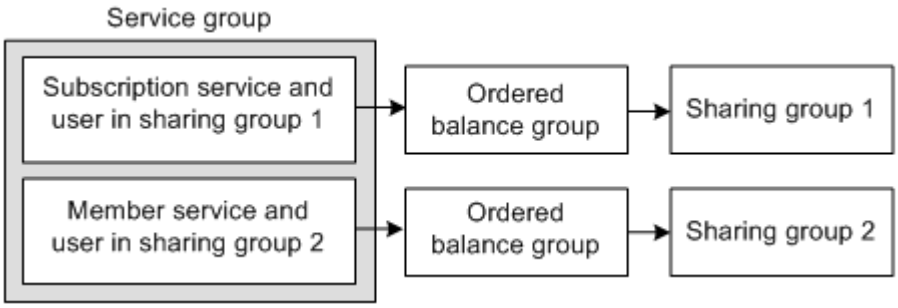
- A member service that has its own ordered balance group does not use the subscription service's ordered balance group. This is true even when the member service and subscription service belong to the same sharing group as shown in Figure 25-3.

Figure 25-3 Member Service Using Its Own Ordered Balance Group



- If a member service and its subscription service belong to different sharing groups, the services have access only to the shared balances in the sharing group to which they belong (see Figure 25-4).

Figure 25-4 Subscription and Member Services in Different Sharing Groups



When the subscription service and a member service are both discount sharing group owners and have members in common, the sequence of the ordered balance group list for each discount sharing group member determines whether the member service's discounts or the subscription service's discounts are applied first.

If a service is a member of both a charge sharing group and a discount sharing group, the discounts are used before charges are applied.

You set up discounts and define how charges are shared in PDC or Pricing Center.

To create charge and discount sharing groups, you implement BRM opcodes in your customer relationship management (CRM) system.

About Changing Status in Service Groups

The status of the subscription service or any member service can be active, inactive, or closed. Changing the status of the subscription service changes the status of all the group's member services.

- When you inactivate a subscription service, all its member services are inactivated.
- When you close a subscription service, all its member services are closed.
- When you reactivate a subscription service that was inactivated or closed, all its member services are reactivated.

Changing the status of a member service affects only that service. If you close a member service and then close the subscription service, reactivating the subscription service does not reactivate the independently closed member service.

Closing a service cancels all charge and discount offers owned by that service. If you reactivate a service and want to restore canceled offers, you must repurchase the offers for the service.

About Setting Up Corporate Accounts That Use Service Groups

When a corporate account uses service groups, rating performance is affected if the company has many employees whose services share account balances.

For corporate accounts, it is better to set up a bill unit hierarchy in which the company account owns the paying parent bill unit and the employees own accounts that include the following:

- Their own subscription and member services
- Nonpaying child bill units of the corporate account's paying parent bill unit

That way, the charges are rolled up to the corporate account at billing time, so rating performance is less affected.

About Creating and Managing Service Groups

Creating and managing service groups includes the following tasks:

- About Creating a Service Group
- About Adding Bundles to Service Groups
- About Setting Up Balance Groups for Service Groups
- About Sharing ERAs in Service Groups
- About Modifying Service Groups

About Creating a Service Group

A subscription service is a **/service** object in the BRM database. You can create a custom storable class to represent a subscription service, or you can use an existing storable class. Member services can be a subclass of the subscription service or a different service type.

A service instance can be either a subscription service or a member service, but not both. After a service is selected as a subscription service, it cannot also be used as a member service in the same service group.

You can create service groups in the following ways:

- When creating a package in PDC or Pricing Center. When customers purchase the package, the member service objects are created and associated with the subscription service object. Additional member services can be added during customer account creation.
- If you implement your own CRM system, customer service representatives (CSRs) can dynamically set up service groups or add member services to an existing service group when creating customers. To create service groups when creating accounts, you implement BRM opcodes in your CRM system. See *BRM Opcode Guide* for more information.

About Adding Bundles to Service Groups

You can add bundles to the subscription service and to member services. A bundle owned by the subscription service is a subscription bundle. That is, the offers in the bundle are available for rating all services in the service group.

You associate bundles with subscription services when you add bundles to packages in PDC or Pricing Center.

A CSR can also add a bundle to a subscription service or a member service that a customer has purchased.

About Setting Up Balance Groups for Service Groups

Although a subscription service can use the account's balance group, tracking usage for a service group when the subscription service has its own balance group is much easier.

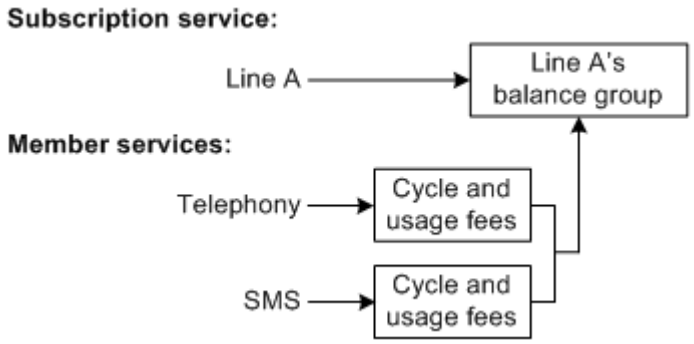
> **Note:**
>
> You cannot transfer a subscription service that uses the account's balance group to another account. See "About Transferring a Service Group to Another Account".

You can do the following:

- **Associate all member services with the subscription service balance group.**

  In this case, when customers use member services, the cycle and usage fees are stored in the subscription service balance group as shown in Figure 25-5.

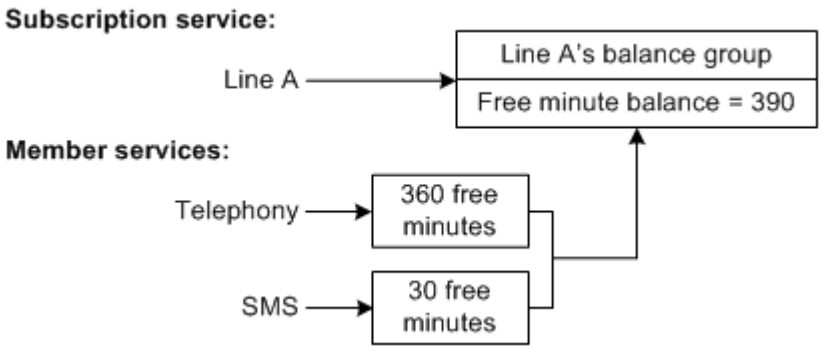
Figure 25-5 Storage of Member Cycle and Usage Fees in Subscription Balance Group



When all member services use the subscription service balance group, all noncurrency balances, such as minutes and frequent flyer miles, are shared by all services in the group.

For example, the Line A subscription service has two member services: a telephony service that includes 360 minutes of usage per month and an SMS service that includes 30 minutes per month. Both member services use the Line A subscription service balance group. At the beginning of the month, a balance of 390 minutes is stored in that balance group, and all services have access to the 390 minutes as shown in Figure 25-6.

Figure 25-6 Sharing of Noncurrency Balances

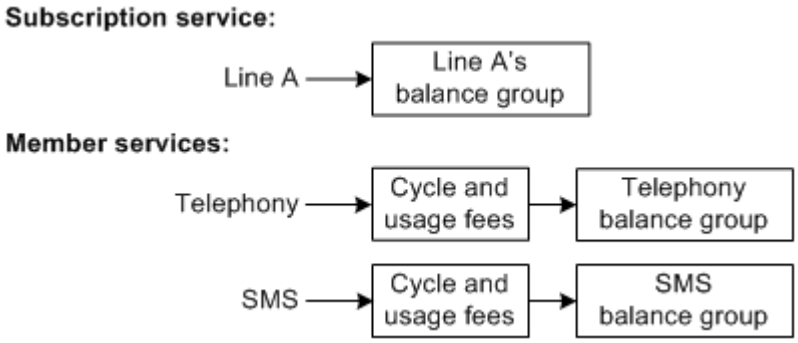


- **Create a balance group for each member service in the group.**

  This makes it possible to set up shared discounts, set credit limits on service balances, and limit balance consumption to the specified service. Initially, the member service balance groups have the same credit limit, floor, and threshold as the subscription service balance group. You can later change them.

  When member services have their own balance groups, cycle and usage fees for each service are tracked in the member service's balance group as shown in Figure 25-7.

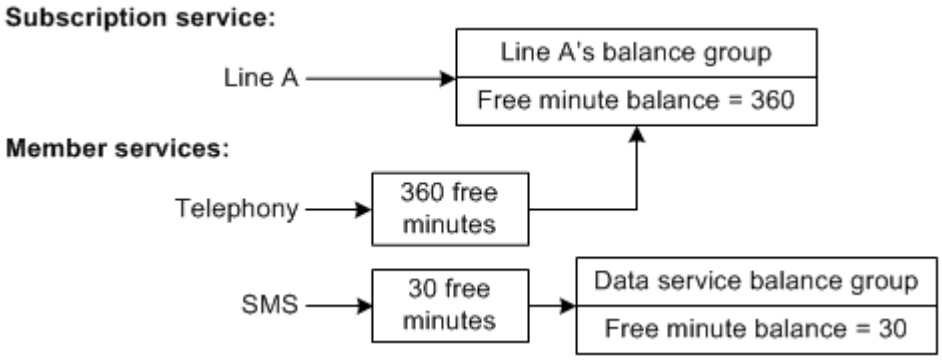
Figure 25-7 Services and Balance Groups



When a member service has its own balance group, any noncurrency balances included with the service can be consumed by that service only.

In Figure 25-8, the SMS service includes 30 minutes that can be applied only to the SMS service, which has access only to its own balances; it cannot use the minutes included with any other service.

Figure 25-8 Noncurrency Services with Multiple Balance Groups



- **Associate groups of services with different balance groups.**

  A member service can also be associated with another member service's balance group to track usage for those services together. However, member services cannot use the balance group of a service that is not in the same service group.

You might want to track some types of noncurrency balances at the subscription level (for example, the number of months users have been subscribers regardless of whether they have changed the services on their lines).

To track noncurrency balances at the subscription level, configure a counter sub-balance for the subscription service. See "About Noncurrency Sub-Balances" in *BRM Creating Product Offerings*.

For more information about balance groups, see *BRM Configuring and Running Billing*.

About Sharing ERAs in Service Groups

Any member service that does not have its own extended rating attribute (ERA) uses the subscription service's ERA. If the subscription service and member service both have an ERA, the member service's ERA is used to rate the member service's usage.

For general information about ERAs, see *BRM Telco Integration*.

About Modifying Service Groups

You can perform the following actions to modify a service group:

- Add a member service to the group.
- Add bundles to a service in the group.
- Change the status of a service in the group.

You can modify a service group's services when setting up services in PDC or Pricing Center.

To modify a service group's services after customers have purchased them, implement BRM opcodes in your CRM system. See *BRM Opcode Guide*.

About Canceling a Subscription Service

A subscription service can be canceled at any time during the billing cycle. You can also bill the account for the service immediately upon cancellation.

In BRM, a subscription service is canceled when the following occur:

- The account that owns the subscription service is closed.
- The subscription service expires.
- The subscription service is canceled by a CSR.

To perform a subscription service cancellation, BRM does the following:

- Closes the subscription service and all its member services. See "About Service Status When a Subscription Service Is Canceled".
- Cancels the charge and discount offers associated with the subscription service and its member services. See "Effect of Canceling a Subscription Service on Offers Owned by the Service".
- Deletes the sharing groups owned by the subscription service and its member services, and removes the services from any sharing groups of which they are members. See "Effect of Canceling a Subscription Service on Sharing Groups Owned by the Service".
- Bills the account if specified. See "About Billing on Subscription Service Cancellation".

CSRs can use Billing Care or Customer Center to cancel subscription services as they cancel other services. To cancel subscription services with your client application, implement BRM opcodes. See "Canceling a Service Group" in *BRM Opcode Guide*.

About Service Status When a Subscription Service Is Canceled

When a subscription service is canceled, all its member services are also canceled. The status of the subscription service and its member services is set to *closed*.

- The status-flag value of the subscription service is set to PIN_STATUS_FLAG_CANCEL_LINE, and the status-flag value of member services is set to PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE, which signifies that the member service status was changed due to the subscription service status change.
- The purchase, usage, and cycle end dates for the subscription service and all its member services are set to the cancellation date.

See "About Changing Status in Service Groups".

Effect of Canceling a Subscription Service on Offers Owned by the Service

Canceling a subscription service cancels all the charge and discount offers owned by the service and its member services.

- The status of the offer instances owned by the subscription service and its member services is set to *canceled*.
- The purchase, usage, and cycle end dates for the canceled offer instances are set to the subscription service cancellation date.

Effect of Canceling a Subscription Service on Sharing Groups Owned by the Service

A subscription service and its member services can own a sharing group, be members of a sharing group, or both.

When a subscription service is canceled, BRM does the following:

- For each sharing group member, the sharing group's balance group is deleted from the group member's ordered balance group list.

- The subscription service and its member services are removed from any sharing groups in which they are members.

For more information, see "About Charge and Discount Sharing Groups".

About Transferring a Service Group to Another Account

You can transfer a service group from one account to another at any time during the billing cycle.

Transferring a service group transfers the subscription service and all member services. The member service objects are updated to reference the target account object.

You can transfer service groups from both closed and active accounts:

- **From a closed account to an active account:** The subscription service remains closed after the transfer. You must manually activate the service.
- **From an active account to a closed account:** The subscription service remains active after the transfer. You must manually activate the account.

All pending scheduled actions (**/schedule** objects) associated with the subscription service and the member services are also transferred to the target account. Completed actions (**/schedule** objects with **Done** status) are not transferred.

To prevent the transfer of a service group to a closed account, use event notification and write a custom opcode to generate an error when the service group transfer event occurs. See "Using Event Notification" in *BRM Developer's Guide*.

> **Note:**
>
> When you transfer a service group to another account, its balance groups are also transferred. If the subscription service or one of its member services uses the account default balance group, however, that balance group cannot be transferred. Therefore, the service group cannot be transferred. To avoid this, the subscription service and member services should use service balance groups. You can define a balance group for the subscription service and associate all the member services with that balance group. See "About Setting Up Balance Groups for Service Groups".

You cannot backdate a service group transfer.

Each time a subscription service is transferred, BRM stores information about the source account and the date and time when the transfer occurred in a *transfer list*. The transfer list is stored in the **/service** object. BRM uses information in the transfer list to rate and record delayed events and call detail records (CDRs) and to bill the source account for usage that occurred before the transfer.

To transfer service groups, you implement BRM opcodes in your client application. See *BRM Opcode Guide* for more information.

About Transferring Service Groups Between Schemas

To transfer a service group between accounts stored in different schemas, do one of the following:

- Migrate the accounts to the same schema and then transfer the service group.

You can use Account Migration Manager (AMM) to migrate accounts between database schemas.

> **Note:**
>
> AMM is an optional feature. Contact your BRM account manager for more information about using AMM.

- If you do not want to migrate the accounts to the same schema, enable the **RecreateDuringSubscriptionTransfer** business parameter. See "Transferring Service Groups between Accounts in Different Schemas" in *BRM Opcode Guide*.

About Transferring Service Groups Associated with Objects

When a service group is transferred to another account, objects associated with the subscription service and its member services are also transferred to that account.

A service can be associated with the following objects:

- Service profiles. See "About Transferring Service Groups Associated with Service Profiles".
- Devices and phone numbers. See "About Transferring Service Groups Associated with Devices".
- Charge and discount offers. See "About Transferring Service Groups Associated with Offers".
- Charge and discount sharing groups. See "About Transferring Service Groups Associated with Sharing Groups ".
- Service sub-balances and account bill units. See "About Transferring Balance Groups During Service Group Transfer".

About Transferring Service Groups Associated with Service Profiles

Service profiles store additional information about a service. A service profile is associated with a service and the account the service belongs to. When a service group is transferred to another account, profile objects associated with the subscription service and its member services are updated to reference the target account.

About Transferring Service Groups Associated with Devices

Devices such as a mobile phone are associated with services. A device object is associated with a service and the account that the service belongs to. When a service group is transferred to another account, the devices associated with the subscription service and its member services are updated to reference that account.

See "Managing Devices with BRM" in *BRM Developer's Guide*.

About Transferring Service Groups Associated with Offers

BRM rates events by using the charge and discount information associated with the service for which the events are generated. The purchase, cycle, and usage start and end dates define the validity periods during which a charge or discount offer can be purchased and cycle fees and usage charges can be applied.

When a service group is transferred, BRM cancels the charge and discount offers associated with the subscription service and its member services for the source account by setting the end dates to the transfer date. When delayed billing is enabled, delayed events and CDRs are rated and discounted by using the canceled charge and discount offers for the source account.

> **Note:**
>
> To rate and discount delayed events for the source account, the canceled charge and discount offers must persist in the account.

Another instance of the charge and discount offers associated with the subscription service and its member services is added to the target account with the purchase, cycle, and usage start dates set to the transfer date. The end dates are set to the original end dates. Events generated by the subscription service after the transfer date are then rated and discounted for the target account.

> **Note:**
>
> - To discount any charges associated with the charge offers, such as the prorated cycle fee amount, the discount offer instances are added to the target account before the charge offer instances.
> - When a service group is transferred, the source and target accounts are not charged any purchase or cancellation fees that are generally applied when a charge or discount offer is purchased or canceled.

See "Managing Purchased Charge Offers".

About Transferring Service Groups Associated with Sharing Groups

Charge and discount sharing groups associated with a subscription service or its member services are not transferred when the service group is transferred to another account. Instead:

- **If a subscription or member service owns a sharing group:** The group is deleted, and the group's balance group is deleted from the ordered balance group list for each member in the group. You must re-create the groups after the service is transferred to the target account.
- **If a subscription or member service is a member of a sharing group:** The service is deleted from the group, and the group's balance group is deleted from the ordered balance group list for the service. The service must be added back to the group after the service is transferred.

See "About Charge and Discount Sharing Groups".

About Transferring Balance Groups During Service Group Transfer

A subscription service can have its own balance group. Each member service can also have its own balance group. The balance group is associated with the account's bill units (**/billinfo** objects).

When a service group is transferred to another account, the balance groups associated with the subscription service and its member services are also transferred to the target account.

Either the balance groups are added to an existing bill unit in the target account, or a new bill unit is created for the service only.

> **Note:**
>
> - You cannot transfer a service group whose subscription service or member services use the account default balance group. To avoid this, subscription and member services should use service balance groups. See "About Creating and Managing Service Groups".
> - Usually, a balance group cannot move to another bill unit when its current bill unit has unallocated payments or adjustments, open refunds, or unresolved disputes. This restriction does not apply, however, when a balance group is moved to another account as the result of a subscription service transfer.

About Assigning Events to Bill Items after a Service Group Transfer

Each event generated by a subscription service is assigned to a bill item (**/item**). During billing, charges from all the bill items are collected in a bill for the account that owns the service group.

When a subscription service is transferred, BRM creates new bill items for the target account by using item configurations (**/config/items**). Events generated after the transfer date are assigned to the new bill items. The new bill items are linked to the bill unit referenced by the service's balance group in the target account. As a result, charges for the new events are applied to the target account.

Delayed events and CDRs are assigned to the bill items in the source account. The bill items are associated with the source account bill unit. Therefore, charges for the bill items are applied to the source account.

About Transferring Noncurrency Assets During Service Group Transfer

Before a service group is transferred to another account, a noncurrency asset is stored in a single sub-balance. After the service group is transferred, each asset is prorated and stored in two sub-balances with different validity periods:

- **Original sub-balance:** By default, assets in this sub-balance are valid from the start of the source account's current billing cycle to the transfer date.
- **New sub-balance:** By default, assets in this sub-balance are valid from the transfer date to the end of the target account's current billing cycle.

> **Note:**
>
> To create separate sub-balances with different validity periods, the balance impact that grants the free assets must have a nonzero relative cycle start date. If the relative cycle offset in the cycle forward or cycle arrears charge is set to 0, a new sub-balance is not created. Instead, the assets are added to the original sub-balance and are valid forever.

Rating and discounting impact the assets in the original sub-balance for delayed events generated by the service before the transfer date. Assets in the new sub-balance are consumed for new events generated by the service group after the transfer date.

By default, the original sub-balance always expires on the transfer date. Because events generated by the target account occur after the transfer date, the target account cannot consume free assets from the original sub-balance. You can extend the validity period of the original sub-balance so that its free assets can be consumed by the target account. See "Extending Sub-Balance Validity When a Service Group Is Transferred".

Extending Sub-Balance Validity When a Service Group Is Transferred

When a service group is transferred to another account, noncurrency assets are prorated and stored in sub-balances with different validity periods. See "About Transferring Noncurrency Assets During Service Group Transfer".

To extend the validity period of the original sub-balance, set the **SubBalValidity** business parameter to one of the following options:

- **Cut:** This is the default. This option sets the validity period from the start of the source account's current billing cycle to the service transfer date.
- **Maintain:** This option retains the original end date of the bucket, even for charge offer cancellations or line transfers.
- **Align:** This option sets the validity period from the start of the source account's current billing cycle to the end of the target account's current billing cycle.

> **Note:**
>
> - Setting the **SubBalValidity** business parameter extends the validity period of the original sub-balance only. The assets in the new sub-balance are always valid from the service transfer date to the end of the target account's billing cycle.
> - Extending the validity period of the original sub-balance extends only the time the free assets in the bucket are accessible and does not change the asset proration. The assets are prorated based on the service transfer date. See "About Prorating Cycle Fees When a Service Group Is Transferred".
> - If you do not configure the **SubBalValidity** parameter, BRM sets the original sub-balance to expire on the service transfer date, and the target account cannot access this sub-balance.

To configure sub-balance validity, see "Configuring Sub-Balance Validity for Service Group Transfer".

About Consuming Noncurrency Balances When a Service Group Is Transferred

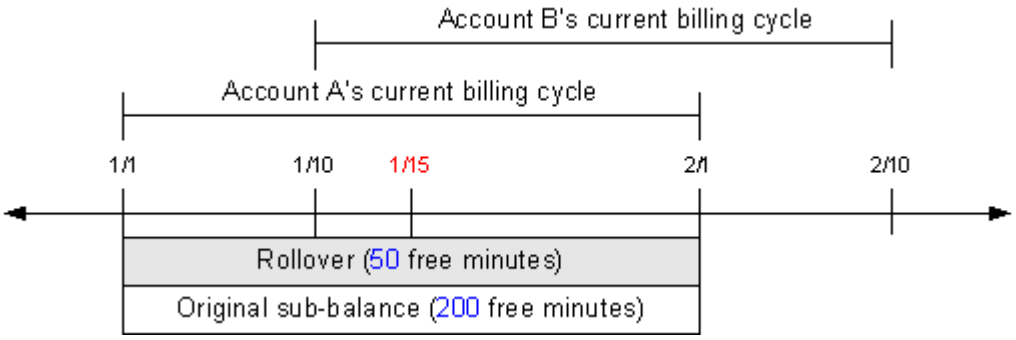
When a service group is transferred to another account, noncurrency balances are prorated and stored in sub-balances with different validity periods. See "About Transferring Noncurrency Assets During Service Group Transfer".

The source account always consumes free balances from the original sub-balance because the events for this account occur before the service transfer date. The target account can consume balances from both the original sub-balance and the new sub-balance when the validity period for the original sub-balance is extended beyond the service transfer date. See "Extending Sub-Balance Validity When a Service Group Is Transferred".

The following examples demonstrate how noncurrency balances are consumed after a service group is transferred.

On January 1, Account A has a rollover balance of 50 minutes and a new grant of 200 minutes. Service Group A is later transferred from Account A to Account B on January 15. Figure 25-9 shows a timeline for the transfer:

Figure 25-9 Example Timeline

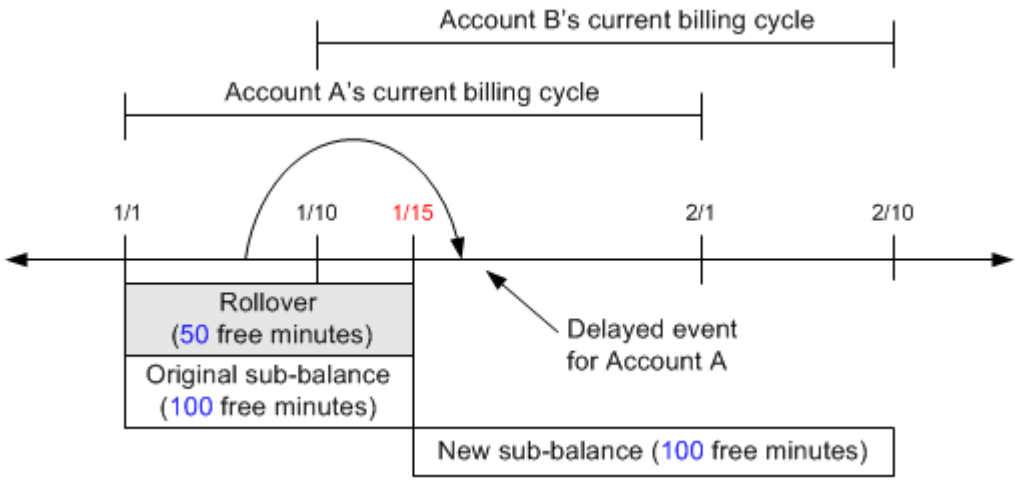


Example 1

When **SubBalValidity** is set to **Cut**, the rollover and original sub-balance buckets' **valid_to** dates are set to the transfer date. The assets in the original sub-balance are prorated, and a new sub-balance is created with 100 minutes.

A delayed event for Account A is recorded on January 17 for 200 minutes. When the event is rated, 50 minutes are consumed from the rollover sub-balance, and 100 minutes are consumed from the original sub-balance. The balance in these two sub-balances becomes 0. Because the event occurred before January 15, it cannot consume minutes from the new sub-balances, and Account A is charged for the remaining 50 minutes of usage. Figure 25-10 illustrates this example.

Figure 25-10 Example 1

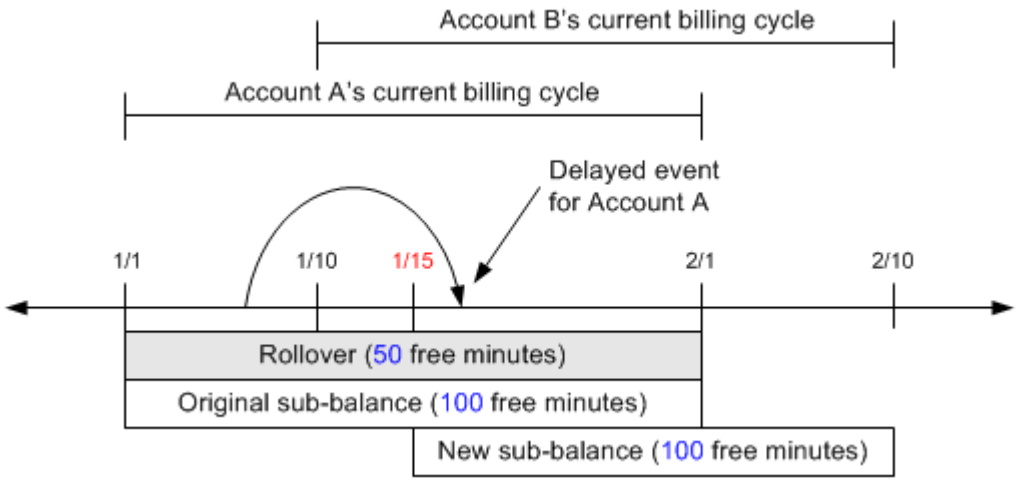


Example 2

When **SubBalValidity** is set to **Maintain,** the rollover and original sub-balance buckets' **valid_to** dates are extended to the end of Account A's billing cycle.

A delayed event for Account A is recorded on January 17 (after the service transfer date) for 200 minutes. When the event is rated, 50 minutes are consumed from the rollover sub-balance, and 100 minutes are consumed from the original sub-balance. The balance in these two sub-balances becomes 0. Because the event occurred before January 15, it cannot consume minutes from the new sub-balance, and Account A is charged for the remaining 50 minutes of usage. Figure 25-11 illustrates this example.

Figure 25-11 Example 2

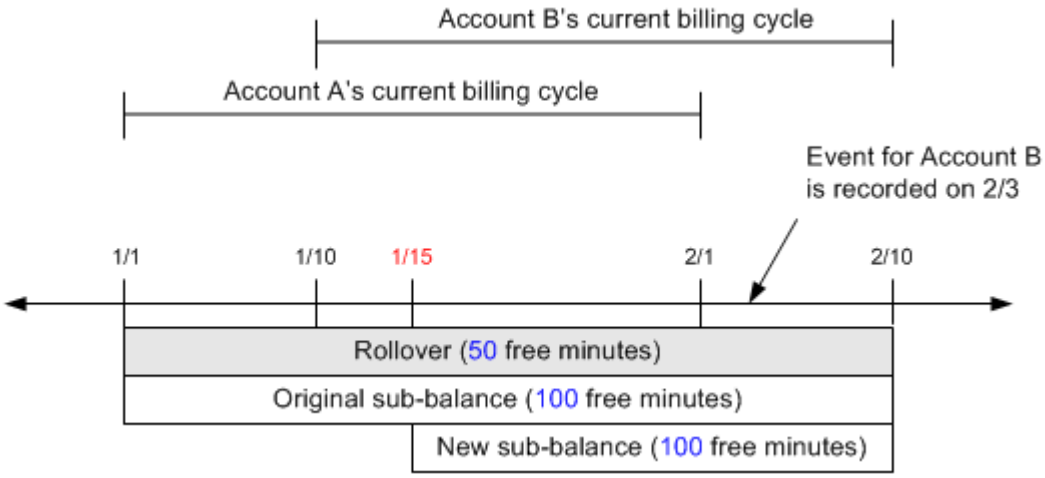


Example 3

When **SubBalValidity** is set to **Align,** the rollover and original sub-balance buckets' **valid_to** dates are extended to the end of Account B's billing cycle.

An event for Account B is recorded on February 3 for 200 minutes. Because the validity period of the rollover and original sub-balance is extended until February 10, 50 minutes are consumed from the rollover sub-balance, and 100 minutes are consumed from the original sub-balance. The remaining 50 minutes are consumed from the new sub-balance. The balance in the rollover and original sub-balance buckets becomes 0, and the balance in the new sub-balance becomes 50. Figure 25-12 illustrates this example.

Figure 25-12 Example 3



About Rolling Over Noncurrency Balances When a Service Group Is Transferred

You can configure rollovers so that noncurrency balances associated with cycle forward fees are rolled over for use in future cycles. See "About Rollovers" in *BRM Setting Up Pricing and Rating*.

When a service group is transferred, the free assets in the rollover and original sub-balance buckets are rolled over when **SubBalValidity** is set to **Maintain** or **Align**.

> **Note:**
>
> The rollover and original sub-balance buckets are not rolled over when **SubBalValidity** is set to **Cut**.

The following example demonstrates how unused balance assets are rolled over when a service group is transferred to another account. In this example:

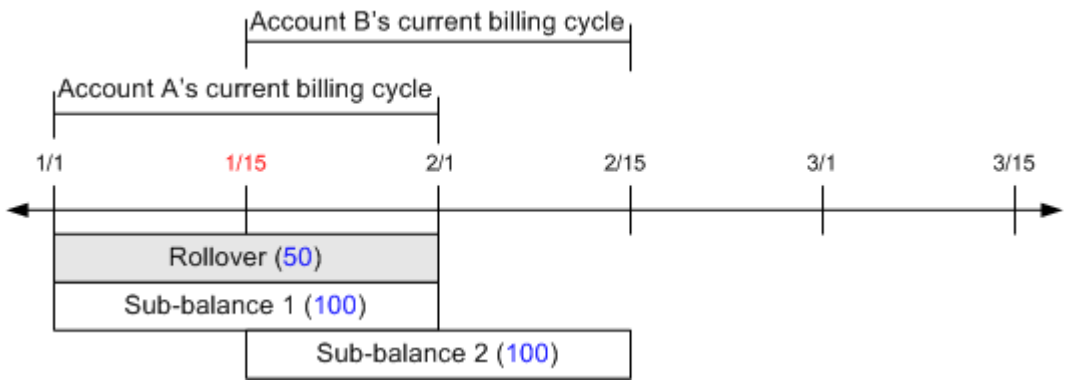
- The service group's subscription service grants 200 minutes valid from January 1.
- The service group is transferred to the target account on January 15.
- The billing cycle for the source account starts on the first of each month.
- The billing cycle for the target account starts on the 15th of each month.
- The **SubBalValidity** business parameter is set to **Maintain**.
- Rollover minutes from the previous cycle are 50.

The rollover rules are as follows:

- A maximum of 50 minutes can be rolled over to the next cycle.
- The maximum number of rollover cycles is 1.
- The maximum number of minutes that can be rolled over from previous months is 100.

On January 1, the source account is granted 200 minutes for the month of January. On January 15, the service group is transferred to the target account. The 200 minutes are prorated and put in two buckets with different validity dates: the original bucket (Sub-balance 1) and a new bucket (Sub-balance 2). Figure 25-13 illustrates this example.

Figure 25-13 Example 4a

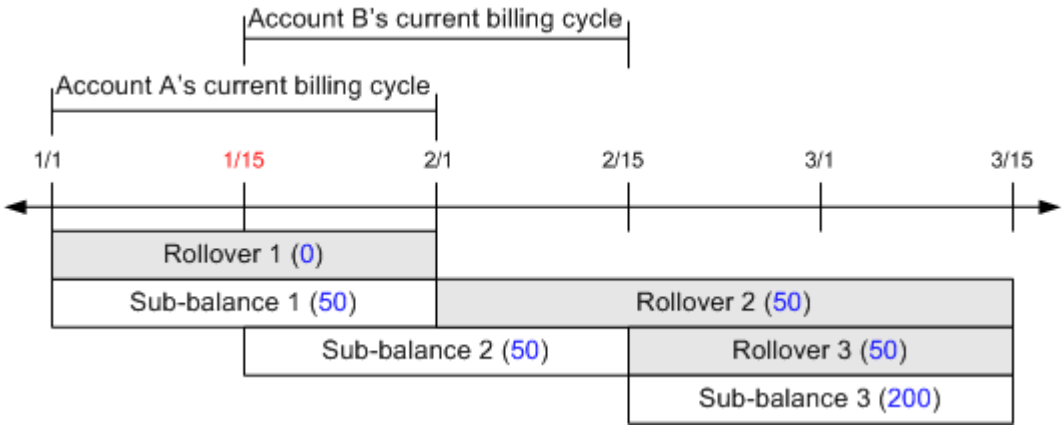


When **pin_bill_day** is run on February 1, 50 minutes from Sub-balance 1 are rolled over. Because Sub-balance 1 is valid for a partial cycle (from January 15 to February 1 instead of January 15 to February 15), the assets are rolled over for one entire cycle. As a result, the Rollover 2 bucket is valid until March 15, which is the next accounting cycle end date for Account B.

When **pin_bill_day** is run on February 15, 50 minutes from Sub-balance 2 are rolled over into a new bucket (Rollover 3) and are valid until March 15. The target account is granted an additional 200 minutes for the new cycle (Sub-balance 3).

Figure 25-14 illustrates the billing timeline associated with this example.

Figure 25-14 Example 4b



Because the Rollover 2 and Rollover 3 buckets are valid after the transfer date, only the target account can access the assets in these buckets. The source account cannot consume the minutes from these buckets.

About Prorating Cycle Fees When a Service Group Is Transferred

Typically, cycle fees are applied at the beginning of a billing cycle for services provided during that cycle and are prorated when a service is purchased or canceled during a billing cycle.

However, a service group transfer is not the same as canceling the charge offer in the source account and purchasing it for the target account. Instead, a service transfer moves the service from one account to another. When a service group is transferred, the accounts are not charged with the charge offer purchase or cancellation fees, and therefore the purchase and cancellation proration generally applied when a charge or discount offer is purchased or canceled are also not applied.

Consequently, when a service group is transferred during a billing cycle, cycle fee proration is enforced based on the transfer date to charge the source account for the fees before the transfer and the target account for the fees after the transfer. The prorated cycle fees from the start of the billing cycle to the transfer date are applied to the source account, and the prorated cycle fees from the transfer date to the end of the billing cycle are applied to the target account.

See "About Proration" in *BRM Configuring and Running Billing*.

Applying Billing-Time Discounts and Folds When a Service Group Is Transferred

Billing-time discounts and folds are applied at the end of the accounting cycle and are based on balances used during the cycle. In BRM, balances are stored and tracked in account default or service balance groups.

When a service group is moved to another account, the service balance group is also transferred to that account. As a result, billing-time folds and discounts are applied to the target account. See "About Transferring Balance Groups During Service Group Transfer".

For more information, see "About Billing-Time Discounts" in *BRM Creating Product Offerings*.

About Rating Delayed Events When a Service Group Is Transferred

Delayed events generated by a service group that occur before the service group transfer are billed to the source account. See "About Assigning Events to Bill Items after a Service Group Transfer".

BRM rates events by using the charge and discount offers associated with the service for which the events are generated. When a service group is transferred, BRM stores the original account's charge and discount offers to rate delayed events generated by that account. See "About Transferring Service Groups Associated with Offers".

When delayed events are rated for the source account, the source account can be charged for usage when all the free assets have been consumed by the target account. This can occur when the validity period of the original sub-balance is extended beyond the service group transfer date. See "About Consuming Noncurrency Balances When a Service Group Is Transferred".

> **Note:**
>
> To rate delayed events accurately, you must rerate events for both the source and target account in the correct order. See "About Rerating Events" in *BRM Rerating Pipeline Events*.

How a Service Group Transfer Affects Account Migration

When an account selected for rerating is associated with a service group transfer, rerating requires both the source account and the target account to reside in the same database schema. For this reason, AMM does not allow the source account or the target account to be migrated to another schema.

For example:

- Account A is stored in Schema A.
- Account B is stored in Schema B.
- Account C is stored in Schema C.

To transfer a service group from Account A to Account B, you must move Account B to Schema A or Account A to Schema B and then transfer the service group to Account B.

After the service transfer, AMM does not allow you to migrate Account A or Account B to Schema C because rerating requires the source account and the target account to reside in

the same schema. To transfer the service to Account C, you must move Account C to Schema A and then transfer the service.

Configuring Sub-Balance Validity for Service Group Transfer

To extend the validity period of the original sub-balance when a service group is transferred, run the **pin_bus_params** utility to change the **SubBalValidity** business parameter. This parameter takes one of the following values:

- **Cut**
- **Maintain**
- **Align**

SubBalValidity is a systemwide parameter. By default, this parameter is set to **Cut**. Changing the value of this parameter affects the sub-balance validity period for any service group transfer that occurs after the change. See "Extending Sub-Balance Validity When a Service Group Is Transferred".

> **Note:**
>
> - If you change the **SubBalValidity** parameter value, BRM does not keep a record of the original setting. BRM uses the new setting for any subscription service transfers that occur after the change.
> - If you do not configure the **SubBalValidity** parameter, BRM follows the default implementation by setting the original sub-balance to expire on the service transfer date.

To extend the validity of the original sub-balance:

1. Go to *BRM_home*/**sys/data/config**.
2. Create an XML file from the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsBilling bus_params_billing.xml
   ```

3. Change **Cut** to one of the following:
   - **Maintain** to extend the validity of the original sub-balance to the end of the original account's current billing cycle.
   - **Align** to extend the validity of the original sub-balance to the end of the target account's current billing cycle.

   ```
   <SubBalValidity>Cut</SubBalValidity>
   ```

4. Save the file as **bus_params_billing.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_billing.xml
   ```

6. Stop and restart the CM.

About Billing for Service Group Usage

To bill for service group usage, all balance groups in the service group must be associated with a bill unit. A bill is produced for each bill unit.

> **Note:**
>
> If a member service's balance group is not associated with a bill unit, charges for that service are not billed.

You can create a bill unit for a service group or use the account's bill unit. If you use the account's bill unit, charges for any service owned by the account that are not part of the service group are also included in the bill.

You use Billing Care or Customer Center to create bill units and associate them with balance groups.

You can also create and manage bill units by using the BRM API. See "Managing Bill Units" in *BRM Opcode Guide*.

About Creating Multiple Bills for a Service Group

When a member service has its own balance group, you can create a separate bill for that service by associating it with a new bill unit. You can also associate several member services with a single bill unit if each member service has a balance group.

About Billing for Multiple Service Groups

When a customer owns multiple service groups, you can bill for each group separately or include charges for all groups in the same bill. To create one bill for all groups, associate the balance groups in all the groups with the same bill unit.

About Billing on Subscription Service Cancellation

Normally, you bill an account on its regularly scheduled billing day at the end of the billing cycle. You can also bill the account immediately upon a subscription service cancellation.

To bill the account immediately, specify the Bill Now option when canceling the subscription service.

> **Note:**
>
> If the Bill Now option is not specified, the account is billed for the subscription service during the regularly scheduled billing time.

When the Bill Now option is specified, the account is billed for *all current* charges generated by the subscription service until the cancellation date.

Delayed events for the service recorded after the cancellation date are billed on the next regularly scheduled billing date.

When the Bill Now option is specified, BRM bills the bill units associated with the subscription service. Typically, all the member services are associated with the subscription service's bill unit. However, a member service can have its own bill unit.

By default, any other service belonging to the same bill unit as the canceled subscription service or its member service's bill unit is also billed.

For more information, see "Configuring Bill Now" in *BRM Configuring and Running Billing*.

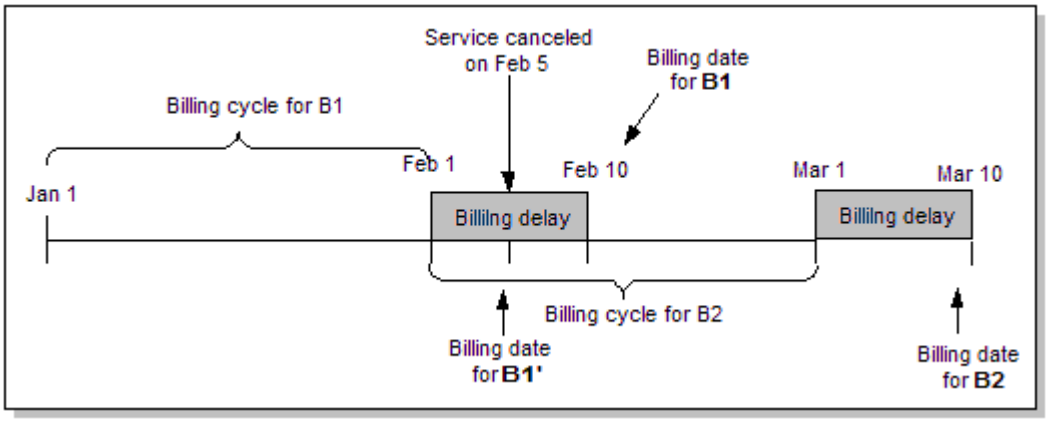
About Billing an Account When a Service Group Is Canceled During the Delayed Billing Period

When a service group is canceled during the billing delay period with the Bill Now option specified, the account is billed immediately.

Delayed events recorded after the cancellation date are billed on the regularly scheduled billing date.

In Figure 25-15, the subscription service is canceled during the billing delay period on February 5. Bill **B1'** includes all current charges from January 1 to February 5 for both billing cycle B1 and billing cycle B2. Delayed events for billing cycle B1 recorded after February 5 are billed on February 10. Events generated for billing cycle B2 after February 5 are billed on March 10.

Figure 25-15 Example 6



About Billing a Service Group After Transferring It to Another Account

A service group can be transferred at any time during the billing cycle.

At the time of billing, any cycle fees associated with the service group are prorated. The source account is billed for the prorated amount until the transfer date. The target account is billed for the remainder of the cycle. See "About Prorating Cycle Fees When a Service Group Is Transferred".

The source account is billed for service usage charges incurred during the time the service was owned by the source account, including delayed events that occurred before the transfer of the service. Events generated after the transfer are billed to the target account.

Any remaining noncurrency grants, such as minutes, are transferred to the target account. See "About Transferring Noncurrency Assets During Service Group Transfer".

26

Creating and Managing Customer Segments

Learn how to use a customer segment, which is a user-defined customer description that can be used to group accounts according to customer billing and payment practices, such as **reliable bill payer**, in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Customer Segments

A customer segment is a user-defined customer description that can be used to group accounts according to customer billing and payment practices, such as **early bill payer**, **reliable bill payer**, and **delinquent bill payer**. You can use customer segments to charge payment fees, provide payment incentives, and suppress bills.

Table 26-1 lists customer segments you can use to implement the following rules for early bill payers who rarely make late payments.

Table 26-1 Customer Segments and Payment Rules

Feature	Payment rule	Action
Payment incentive	If the payment is received 2 weeks early.	Apply $2 discount to the bill.
Payment fee	If the payment fails due to expired credit card.	Override the $2 payment fee.
Bill suppression	If the bill is less than $10.	Suppress the bill until the end of the next billing cycle. **Note:** To implement automatic bill suppression, you must first define customer segments and then associate bill suppression settings with them. See "About Bill Suppression" in *BRM Configuring and Running Billing*.

An account can belong to more than one customer segment.

Defining Customer Segments

You configure customer segments in the *BRM_home***/sys/data/config/pin_customer_segment.xml** file, where *BRM_home* is the directory in which the BRM server software is installed.

To create a customer segment, add a **CustomerSegment** child element to the **CustomerSegments** element. Each **CustomerSegment** child element is identified by an ID and a character string. For example:

```
<CustomerSegmentConfiguration>
     <CustomerSegments>
          <CustomerSegment ID="1001">Early bill payer</CustomerSegment>
          <CustomerSegment ID="1002">Reliable bill payer</CustomerSegment>
          <CustomerSegment ID="1003">Late bill payer</CustomerSegment>
     </CustomerSegments>
</CustomerSegmentConfiguration>
```

The ID and string items are described in Table 26-2.

Table 26-2 XML Items in pin_customer_segment.xml File

XML items	Description	Possible values
ID	A number that identifies the customer segment in the BRM database. An account belongs to a customer segment when that segment's ID is added to the **/account** object's PIN_FLD_CUSTOMER_SEGMENT_LIST field.	Specify any integer greater than **1000.** **Note:** If a customer segment is not defined for an account, the default value of **0** is used. All accounts belong to this customer segment.
string	A character string that describes the type of accounts in the customer segment (for example, reliable bill payer or delinquent bill payer).	Minimum length is 0 characters. Maximum length is 1023 characters. **Note:** This string is mapped to the PIN_FLD_DESCR field in the **/config/customer_segment** object.

Run the "load_pin_customer_segment" utility to load the contents of the **pin_customer_segment.xml** file into the **/config/customer_segment** object in the BRM database.

> **Note:**
>
> - The **load_pin_customer_segment** utility needs a configuration file (**pin.conf**) in the directory from which you run the utility. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.
> - The **load_pin_customer_segment** utility overwrites existing customer segments. If you are updating customer segments, you cannot load new customer segments only. You must load complete sets of customer segments each time you run the **load_pin_customer_segment** utility.

To load customer segments into BRM:

1. Open the **pin_customer_segment.xml** file (*BRM_home***/sys/data/config**) by using an XML editor or a text editor.
2. Edit the file.
3. Save the file.

> **Note:**
>
> Customer segment IDs in the 0 - 1000 range are reserved by BRM. The default customer segment ID is **0**, and all accounts belong to this customer segment by default.

4. Use the following command to load the customer segments:

```
load_pin_customer_segment pin_customer_segment.xml
```

> **Note:**
>
> The **pin_customer_segment.xml** file is referenced by the **pin_business_configuration.xsd** file. Therefore, these files must be located in the same directory. By default, the location is *BRM_home*/**sys/data/config**.

If you do not run the utility from the directory in which the XML and XSD files are located, include the complete path to the files. For example:

```
load_pin_customer_segment BRM_home/sys/data/config/pin_customer_segment.xml
```

5. Stop and restart the Connection Manager (CM).
6. (Optional) To verify that the customer segments were loaded, display the **/config/customer_segment** object by using Object Browser or by using the **robj** command with the **testnap** utility. See "Reading an Object and Writing its Contents to a File" in *BRM Developer's Guide*.

To validate the XML schema in your file, run the "load_pin_customer_segment" utility with the **-t** parameter.

This parameter validates the contents against the XML file's schema definition before loading the data into the **/config/customer_segment** object.

> **Note:**
>
> The schema definition for the **pin_customer_segment.xml** is in the *BRM_home*/**xsd/pin_customer_segment.xsd** file. This file uses the **pin_business_configuration.xsd** reference file to perform additional processing.

Implementing Customer Segment Information

To implement customer segments, create your own application or use a third-party client application to assign customer segments during account maintenance.

After you define customer segments in BRM, any rules you define for a segment apply to all accounts that belong to that customer segment. For example, if you define payment fees or payment incentives based on a customer segment, all accounts that belong to that segment, and conform to any additional criteria, will receive the fee or incentive.

To add customer segments to an account during account maintenance, include them in the PCM_OP_CUST_UPDATE_CUSTOMER input flist. The segments are in the PIN_FLD_CUSTOMER_SEGMENT_LIST field in the PIN_FLD_ACCTINFO array.

> **Note:**
>
> Accounts created by a custom application are assigned a customer segment value of **0**.

Part VII
Managing Customer Data

This part provides information about managing your customers' data in Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Creating and Managing Business Profiles
- Recording Login Failures
- Masking Sensitive Customer Data
- Validating Account Data

27

Creating and Managing Business Profiles

Learn how to use business profiles to classify types of data to specify how to treat the data in Oracle Communications Billing and Revenue Management (BRM). For example, business profiles can be used to classify bills as prepaid or postpaid to determine how bills and payments should be handled.

Topics in this document:

About Business Profiles

Use business profiles to classify types of data to specify how to treat the data. For example, business profiles can be used to classify bills as prepaid or postpaid to determine how bills and payments should be handled.

A business profile includes the following:

- A set of requirements that a bill unit and its related objects must meet to be associated with the profile. These requirements are stored in validation templates linked to the business profile. You can create validation templates for the following objects:
  - **/account**
  - **/balance_group**
  - **/billinfo**
  - **/group/sharing/charges**
  - **/group/sharing/discounts**
  - **/group/sharing/profiles**
  - **/ordered_balgrp**
  - **/profile/acct_extrating**
  - **/profile/serv_extrating**
  - **/purchased_discount**
  - **/purchased_product**
  - **/service**
- An array of key-value pairs used to retrieve information about the bill units that belong to the business profile. For example, a business profile object that includes the key **Prepaid** with the value **Yes** indicates that bill units belonging to the business profile are prepaid. A business profile set up for postpaid bill units includes the key **Postpaid** with the value **Yes.** To find out whether a bill unit is associated with a prepaid or postpaid bill, use the PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO opcode to check the value of this key in the bill unit's business profile. See "Getting Information about a Business Profile" in *BRM Opcode Guide* for information.

> **Note:**
>
> To use business profiles, create a custom user interface (UI) that does the following:
>
> - Enables customer service representatives (CSRs) to assign bill units to business profiles.
> - Provides different interfaces based on the business profile of a bill unit. For information about setting up business profiles and assigning bill units to them, see "Creating and Managing Business Profiles".

A bill unit can belong to only one business profile at a time. To assign a bill unit to a business profile, BRM puts the Portal object ID (POID) of the business profile object into the PIN_FLD_BUSINESS_PROFILE_OBJ field of the **/billinfo** object.

Before assigning a bill unit to a business profile, BRM verifies that the bill unit and all of its associated objects comply with the requirements specified in the business profile's validation templates.

To create business profiles and validation templates, you edit the *BRM_home***/sys/data/config/pin_business_profile.xml** business profile configuration file, where *BRM_home* is the directory in which the BRM server software is installed. This file contains three default business profiles that can be used to set up cache residency distinction. See "Editing the Business Profile Configuration File".

After editing the configuration file, you use the "load_pin_business_profile" utility to load the file's contents into the BRM database. See "Setting Up Business Profiles and Validation Templates".

Setting Up Business Profiles and Validation Templates

To create, modify, or delete business profiles (**/config/business_profile** objects) and validation templates (**/config/template** subclass objects), edit the business profile configuration file (**pin_business_profile.xml**) and then load its contents into the BRM database:

1. Open the **pin_business_profile.xml** file in an XML editor or a text editor.

   By default, the file is in the *BRM_home***/sys/data/config** directory.
2. Enter the appropriate information into the file. See "Editing the Business Profile Configuration File".
3. Save and close the file.
4. Use this command to load **pin_business_profile.xml** file:

   ```
   load_pin_business_profile pin_business_profile.xml
   ```

> **Note:**
>
> - When you run the utility, the **pin_business_profile.xml** and **business_configuration.xsd** files must be in the same directory. By default, both files are in *BRM_home***/sys/data/config**.
> - This utility needs a configuration (**pin.conf**) file in the directory from which you run the utility.
> - If you do not run the utility from the directory in which **pin_business_profile.xml** is located, include the complete path to the file. For example:
>
>   ```
>   load_pin_business_profile BRM_home/sys/data/config/
>   pin_business_profile.xml
>   ```

5. To verify that the business profile and validation template information was loaded, display the **/config/business_profile** objects and **/config/template** subclass objects by using one of the following features:
   - Object Browser
   - **robj** command with the **testnap** utility

   For information about reading an object and writing its contents to a file, see "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Editing the Business Profile Configuration File

You configure all of the business profiles (**/config/business_profile** objects) and validation templates (**/config/template** subclass objects) in your BRM system in the *BRM_home***/sys/data/config/pin_business_profile.xml** file. This file contains prepaid, postpaid, and convergent business profiles by default.

To edit this configuration file, open it in an XML editor or text editor and then perform these tasks:

- Defining Business Profiles
- Modifying Business Profiles
- Deleting Business Profiles
- Deleting Validation Templates

Defining Business Profiles

In the business profile configuration file, business profiles (**/config/business_profile** objects) are defined as **<BusinessProfile>** child elements of the **<BusinessProfileList>** parent element. A **<BusinessProfile>** child element consists of the following elements and attributes:

- The name of the business profile (BusinessProfile name).
- An optional attribute (type). If the **type** attribute value is **Invoice**, the business profile is identified as an invoicing business profile. See "Creating an Invoicing Configuration Business Profile" in *BRM Designing and Generating Invoices*.
- A description of the business profile (Desc).
- A list of associated validation templates (TemplateId).

- A list of key-value pairs (NameValue). An unlimited number of key-value pairs can be associated with a business profile. See "About Business Profiles".

The syntax of a **<BusinessProfile>** child element is:

```
<BusinessProfileList>
 <BusinessProfile name="string" type="string">
  <Desc>string</Desc>
  <!-- Specify List of Templates -->
  <TemplateId name="string" type="object_type"/>
  <TemplateId name="string" type="object_type"/>
  <TemplateId name="string" type="object_type"/>
  <!-- Specify List of Key Values -->
  <NameValue key="string" value="string"/>
 </BusinessProfile>
</BusinessProfileList>
```

To create a business profile, add a **BusinessProfile** child element to the **BusinessProfileList** parent element. In the child element, specify values for the items listed in Table 27-1.

Table 27-1 Business Profile Elements

XML element or attribute	Description	Possible values
BusinessProfile name	Character string used as the name of the business profile object (for example, **PrepaidGold**).	The name must be unique within your BRM system. Minimum length is 1 character. Maximum length is 255 characters. **Note:** This string is mapped to the PIN_FLD_NAME field in the **/config/business_profile** object. Values in this field can be used to populate a list of business profiles in a user interface (UI). When creating the string, take any UI length restrictions into consideration.
type	An optional attribute that identifies a business profile.	Invoice
Desc	Character string that describes the type of bill units that belong to the business profile (for example, **Bill units with credit balances**).	Minimum length is 1 character. Maximum length is 255 characters. **Note:** This string is mapped to the PIN_FLD_DESCR field in the **/config/business_profile** object. Values in this field can be used to populate a list of business profile descriptions in a UI. When creating the string, take any UI length restrictions into consideration.
TemplateId	ID of a validation template associated with the business profile. In each **TemplateId** element, specify the following: • The **name** value of a validation template defined in the file's **TemplateList** element. • The **type** value of the same template.	-

Table 27-1 (Cont.) Business Profile Elements

XML element or attribute	Description	Possible values
NameValue	Key-value pair used to retrieve information about objects associated with the business profile. In each **NameValue** element, specify the following: • **key**: Character string used with **NameValue value** to describe characteristics of the bill units belonging to the business profile. • **value**: Character string used with **NameValue key** to describe characteristics of the bill units belonging to the business profile (for example, **Yes** and **No**).	Minimum length of each character string is 1 character. Maximum length of each character string is 255 characters.

Modifying Business Profiles

To modify a business profile already defined in your BRM system, redefine the business profile in your business profile configuration file (see "Defining Business Profiles"). You can change any aspect of a business profile except "BusinessProfile name".

When you load the contents of the configuration file into your BRM database (see "Setting Up Business Profiles and Validation Templates"), BRM incorporates your changes into the existing business profile object.

Deleting Business Profiles

To delete a business profile from your BRM system, add the following child element to the **<BusinessProfileList>** element in your business configuration file:

```
<BusinessProfile name="name_of_profile_to_delete" action="delete">
```

> **Note:**
>
> You cannot delete a business profile to which any bill unit belongs.

When you load the contents of the configuration file into your BRM database (see "Setting Up Business Profiles and Validation Templates"), BRM deletes the specified business profile.

Deleting Validation Templates

To delete a validation template from your BRM system, add the following child element to the **<TemplateList>** element in your business configuration file:

```
<Template name="name_of_template_to_delete" type="type_of_template_to_delete"
action="delete">
```

> **Note:**
>
> You cannot delete a validation template associated with a business profile.

When you load the contents of the configuration file into your BRM database (see "Setting Up Business Profiles and Validation Templates"), BRM deletes the specified validation template.

28

Recording Login Failures

Learn how to record login failures in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Recording Login Failures

You can record login failures to detect possible service problems or fraud. For example, duplicate logins might indicate fraudulent usage. You can specify which login failures to record.

You can record login failures for any reason except using an unknown login name. For example, you can record events when customers try to:

- Log in with the wrong password
- Log in to an inactive service
- Log in after exceeding their credit limit

You can also record successful logins, although doing this slows BRM performance.

To record login failures, specify the types of login failures you want to record in the **pin_verify** file and then load those preferences into the BRM database with the **load_pin_verify** command.

> **Note:**
>
> The **load_pin_verify** utility requires a configuration file. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

To specify login failure types:

1. Edit the *BRM_home*/**sys/data/config/pin_verify** file, where *BRM_home* is the directory in which the BRM server software is installed. The **pin_verify** file includes examples and instructions.

   You can make these types of changes:

   - Edit a list of predefined types of login failures to specify which events you want to record and to modify their descriptions.
   - Add custom types of login failures you want to record.

> **Note:**
>
> When you run **load_pin_verify**, it overwrites the existing preferences for recording customer login failures. If you are updating your preferences, you cannot load new preferences only. You must load complete sets of preferences each time you run the **load_pin_verify** utility.

2. Save the file.
3. Use the following command to run **load_pin_verify** utility:

   ```
   load_pin_verify pin_verify_file
   ```

   For more information, see "load_pin_verify".
4. Stop and restart the CM.

To verify that the network elements were loaded, you can display the **/config/verify** object by using Object Browser, or by using the **robj** command with the **testnap** utility. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Specifying the Account That Records Login Failures

BRM logs verification events against the **root** account by default. If you want to use a different account, do the following:

1. (Optional) Create an account specifically for logging verification events. You can create a CSR account or a dummy account.

   If you create a dummy account, you should:

   - Use a dummy package that includes no charge pricing and applies to accounts but not to any service.
   - Use the internal payment method.
   - Use any name and address you want. You have to enter something for name and address to create an account.
2. Open the CM configuration file (*BRM_home***/sys/cm/pin.conf**).
3. Add this entry to the end of the file:

   ```
   - fm_act account_name database_number /account 1
   ```

   where:

   - *account_name* is the name of the account to which verification logging should go.
   - *database_number* is the database number of the BRM database. By default, this number is 0.0.0.1.

   This example sets an account called **verify_acct** to log verification events:

   ```
   - fm_act verify_acct 0.0.0.1 /account 1
   ```
4. Save the file.
5. Stop and restart the CM.

Viewing Login Failures

To view login failures, use Billing Care or Customer Center to open the root account or the account you specified in the CM configuration file.

29

Masking Sensitive Customer Data

Learn how to mask sensitive customer data in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Masking Sensitive Customer Data

You can prevent access to and logging of sensitive customer data, such as banking information and passwords, by masking the string data fields that store this information. BRM client applications, transaction messages, and logs can contain sensitive information.

> **Note:**
>
> By default, data masking is enabled to protect sensitive data.

Masking Sensitive Data in Logs When Using Client Applications

Portal Communication Module (PCM) C++ and PCM Java client applications log flists containing sensitive customer data before calling the CM for processing. Client application logs can contain flists in either the standard BRM format or in XML format. Standard format flist fields configured for masking are automatically hidden before logging by the PIN_FLIST_TO_STR macro. To mask sensitive data in flists stored in XML format during logging, your client application must call the **XMLToFlist** class included in the **pcm.jar** file.

For more information on developing PCM C++ and Java client applications see "Creating Client Applications by Using PCM C++" and "Creating Client Applications by Using Java PCM" in *BRM Developer's Guide*.

Masking Additional Fields in Client Application Logs

BRM masks the following string data fields in client application logs for standard format flists and XML flists processed by the **XMLToFlist** class:

- PIN_FLD_PASSWD
- PIN_FLD_PASSWD_CLEAR
- PIN_FLD_DEBIT_NUM
- PIN_FLD_DEBIT_EXP
- PIN_FLD_BANK_NO
- PIN_FLD_BANK_ACCOUNT

- PIN_FLD_BANK_ACCOUNT_NO
- PIN_FLD_IBAN

You can add additional fields to be masked, including custom fields, in client application logs based on your security requirements. A list of default BRM fields are defined in the **pin_flds.h** file while custom fields are found in the **cust_flds.h** file. See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide* for information on adding custom fields.

To mask additional fields in logs of PCM C++ applications:

1. Open the *BRM_home***/include/pin_flds.h** file or the *BRM_home***/include/cust_flds.h** file and obtain the field IDs you want to mask.
2. For each field requiring masking, add a line at the end of either the **pin_flds.h** or **cust_flds.h** file using the following syntax:

   ```
   #define Field_Name PIN_MAKE_FLD(Field_Name,ID)
   Field_Name masked
   ```

   where *Field_Name* is the string field to mask in PCM C++ client application logging, and *ID* is the BRM assigned ID for the field. For example, mask the PIN_FLD_CHECK_NO field by using the following line:

   ```
   #define PIN_FLD_CHECK_NO PIN_MAKE_FLD(PIN_FLDT_STR, 931)
   PIN_FLD_CHECK_NO masked
   ```

   > **Note:**
   >
   > Ensure that your **cust_flds.h** file does not contain duplicate entries.

3. Save the file.
4. If only masking additional fields in **pin_flds.h**, restart the CM.

To mask fields defined in **cust_flds.h**, complete the following additional steps:

1. Make a copy of your **cust_flds.h** file named **masked_fields**.
2. Run the *BRM_home***/bin/parse_custom_ops_fields.pl** perl script which generates the **masked_fields.dat** file using the following syntax:

   ```
   perl -S parse_custom_ops_fields.pl -L pcmc -I masked_fields -O
   masked_fields.dat
   ```

3. Add the following entry in the **pin.conf** file for your PCM C++ client application:

   ```
   - - ops_fields_extension_file path/masked_fields.dat
   ```

   where *path* is the path to the **masked_fields.dat** file.

   > **Note:**
   >
   > The **pin.conf** file must only have one **-- ops_fields_extension_file** entry. If you already have a **cust_flds.h** file, append your masking entries in the same file and generate a single **masked_fields.dat** file.

4. Restart the CM.

To add additional string data fields for masking in logs of PCM Java applications:

1. Open the *BRM_home*/**include/pin_flds.h** file or the *BRM_home*/**include/cust_flds.h** file and obtain the field IDs. BRM assigns each field a numerical value listed at the end of each row. Use this field ID in the **Infranet.properties** file for masking. For example, the field ID for the PIN_FLD_CHECK_NO field is **931**:

   ```
   #define PIN_FLD_CHECK_NO PIN_MAKE_FLD(PIN_FLDT_STR, 931)
   ```

2. Open the **Infranet.properties** file for your PCM Java application in a text editor.
3. Add a line for each additional field to be masked using the following syntax:

   infranet.custom.masked.field.*field_id*=**masked**

   where *field_id* is the field ID for the default or custom field you want to mask.
4. Save the file.
5. Verify that the **Infranet.properties** file is included in the CLASSPATH of the PCM Java client application process.

30
Validating Account Data

Learn how to analyze data from the Oracle Communications Billing and Revenue Management (BRM) database.

Topics in this document:

- About Validating Account Information

About Validating Account Information

The Account Dump utility (ADU) is a diagnostics tool that enables you to validate account information before or after certain business processes, for example, after completion of a migration or upgrade or before billing or payment allocation.

> **Note:**
>
> ADU can be performance intensive, depending on the number of accounts for which data is being retrieved and the volume of data being processed. Avoid running performance-intensive operations, such as billing, while running ADU.

The account dump and data validation process is a follows:

1. Create an input file with the account search specification. ADU uses the specification to select accounts in the BRM database. See "About ADU Account Search".
2. ADU searches the accounts in the BRM database, retrieves the related object information, and dumps the information to an output file. See "About ADU Account Dump".
3. ADU validates the contents of the output file. See "About Account Data Validation".

About ADU Account Search

You can request ADU to dump information for a single account or for multiple accounts. To specify the accounts for which you want to dump information, you provide as input a text file that contains the account search specification. The search specification must be in the form of an flist.

For example, the following search flist requests the dump of the account **0.0.0.1 /account 789888 0**:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_RESULTS        ARRAY [0]
1    PIN_FLD_ACCOUNT_OBJ  POID [0] "/account" 789888 0
```

The following search flist requests the dump of all the accounts with billing day of month (DOM) as 1:

```
0 PIN_FLD_POID      POID [0] 0.0.0.1 /search/pin -1 0
0 PIN_FLD_FLAGS     INT [0] 256
```

```
0 PIN_FLD_TEMPLATE  STR [0] "select X from /billinfo where F1 = V1 "
0 PIN_FLD_RESULTS   ARRAY [0]
1    PIN_FLD_ACCOUNT_OBJ POID [0] NULL
0 PIN_FLD_ARGS      ARRAY [1]
1    PIN_FLD_ACTG_CYCLE_DOM INT [0] 1
```

Note:

- ADU considers each account as a standalone. If a group owner account is specified in the search flist, ADU dumps only the contents of the owner account. ADU will not dump the contents of the group member accounts. Similarly, if a group member account is specified in the search flist, only the contents of the member account is dumped. ADU will not dump the contents of the owner account or other group member accounts.
- The PIN_FLD_RESULTS array in the search flist must contain only PIN_FLD_ACCOUNT_OBJ.
- ADU can be performance intensive, depending on the number of accounts for which data is being retrieved and the volume of data being processed. Run ADU in **-report** mode first to determine the volume of data to be processed so that you can optimize your account search for best performance. See "pin_adu_validate" in *BRM Developer's Guide*.

About ADU Account Dump

ADU provides the flexibility to choose the object information that you want dumped for an account. For example, you can request ADU to dump the **/account**, **/service**, and **/invoice** object information only.

You can also select the fields of an object that you want dumped. ADU then dumps only those fields into the output file. For example, you can request to dump only the first and last names of the account.

Some objects, such as **/audit** and **/event,** contain large volumes of data. Searching and retrieving information from these objects can be system intensive. Therefore, ADU provides the option to select data from these objects by specifying a date range. For example, you can configure ADU to dump the **/event** objects of an account updated between February 1st and March 1st. For more information on using the date range option, see "Limiting Dump Information by Specifying a Date Range".

A separate output file is generated for each account. ADU uses the format **account**_*POID_ID*0.*File_Extension* to generate the output file name where *POID_ID* is the BRM account object identifier and *File_Extension* is the extension defined in the ADU configuration file. The file extension can be configured in the ADU configuration file (**pin.conf**). See "Configuring ADU".

Note:

The output file is overridden if the dump is requested for the same accounts more than once and is stored in the same output folder.

By default, ADU dumps the object contents in the output file in XML format. If you prefer a different output format (for example, CSV), you can customize the output format by modifying the PCM_OP_ADU_POL_DUMP policy opcode.

Limiting Dump Information by Specifying a Date Range

You can limit the dump for the following objects in the BRM database by specifying a date range:

- **/audit**
- **/bill**
- **/item**
- **/event**
- **/invoice**
- **/balance_group**

These objects normally contain large volumes of data. To limit the amount of data retrieved from these objects, use the date range option to select only data updated during a specified period.

For example, if you choose to dump the contents for **/account**, **/service**, **/bill**, and **/item** and specify February 1, 2007, as the **dump_start_time** and March 1, 2007, as the **dump_end_time**, ADU uses that date range for searching and retrieving data from the **/bill** and **/item** objects only. The date range is not used for retrieving data from the **/account** and **/service** objects.

The date range is mapped to the object date fields as follows:

- For the **/audit** object, ADU selects only those objects whose **created_t** or **effective_t** is between **dump_start_time** and **dump_end_time**.
- For the **/bill** object, ADU selects only those objects whose **end_t** is between **dump_start_time** and **dump_end_time**.
- For the **/item** object, ADU selects only those objects whose **effective_t** is between **dump_start_time** and **dump_end_time**.
- For the **/event** object, ADU selects only those objects whose **end_t** is between **dump_start_time** and **dump_end_time**.
- For the **/invoice** object, ADU selects only those objects whose **created_t** is between **dump_start_time** and **dump_end_time**.
- For the **/balance_group** object, ADU selects only those objects whose **effective_t** is between **dump_start_time** and **dump_end_time**.

To configure the date range, see "Configuring ADU".

About Account Data Validation

ADU performs the following types of validations on object contents:

- **Structural validation**: Validates the structure of the account. For example, it validates that the POID of the parent **/bill** object exists in the nonpaying child **/bill** object.
- **Dynamic validation**: Validates the business logic. For example, it validates that the **/item** due amount is zero after a payment.

Table 30-1 contains the predefined structural and dynamic validations. Each validation is associated with a validation code.

Table 30-1 Predefined Structural and Dynamic Validations

Validation Type	Validation Code	Description
Structural	struct_valid_01	Validates that **/event** objects point to the correct **/item** objects based on the mappings configured in **/config/item_tags** and **/config/item_types**.
Structural	struct_valid_02	Validates that the PIN_FLD_PARENT field of the nonpaying child bill unit's last bill points to the parent **/bill** object. Validates that the bill number of the nonpaying child bill unit and the bill number of the parent bill unit are the same.
Structural	struct_valid_03	Validates that the billing DOM of the nonpaying child bill unit and the billing DOM of the paying parent bill unit are the same.
Structural	struct_valid_04	Validates that the AR_BILLINFO_OBJ of the nonpaying child bill unit points to the **/billinfo** object of the paying parent bill unit.
Dynamic	dynamic_valid_01	Validates that the **/item** due amount is zero after payment.

You enable these validations by setting the corresponding validation code in the ADU **pin.conf** file. ADU logs the results of the validations in the Connection Manager (CM) log file.

You can customize additional validations by modifying the PCM_OP_ADU_POL_VALIDATE policy opcode. See "Customizing the Account Dump Utility (ADU)" in *BRM Opcode Guide*.

Configuring ADU

To configure ADU, set the entries in the ADU configuration file (*BRM_home***/sys/diagnostics/pin_adu_validate/pin.conf**) shown in Table 30-2.

Table 30-2 Entries in the ADU Configuration File

Entry	Description
- pin_adu input_file *file_name*	Set *file_name* to the name of the input file that contains the account search specification. For example: **- pin_adu input_file** *BRM_home***/sys/diagnostics/pin_adu_validate/in/input.txt**
- pin_adu output_file *file_name*	Set *file_name* to the name of the output folder where ADU should write the output file. For example: **- pin_adu output_file** *BRM_home***/sys/diagnostics/pin_adu_validate/out**
- pin_adu out_file_ext.*ext*	Set *ext* to the output file extension. For example: **- pin_adu out_file_ext.xml**
- pin_adu obj_list *object1***;** **[***object2***];** ...	Specify the objects to dump for the selected accounts. For example: To dump **/billinfo** and **/payinfo** objects: **- pin_adu obj_list /billinfo; /payinfo** **Note:** Use a semicolon to separate items in the object list.

Table 30-2 (Cont.) Entries in the ADU Configuration File

Entry	Description
- pin_adu obj_flds *object1*:*field1*, *field2*, ... **[***object2*:*field1*, *field2*, ...**]** ...	Specify the fields in the objects to dump. For example: To dump the first and last name of an account: **- pin_adu obj_flds /account: PIN_FLD_NAMEINFO.PIN_FLD_FIRST_NAME, PIN_FLD_NAMEINFO.PIN_FLD_LAST_NAME** To dump account invoice information: **- pin_adu obj_flds /payinfo/invoice: PIN_FLD_INV_INFO** **Note:** Use a colon to separate the items in the object list.
- pin_adu dump_start_time *time* **- pin_adu dump_end_time** *time*	Set *time* in these entries to the times to use for selecting most commonly updated objects. **- pin_adu dump_start_time 1175385600** **- pin_adu dump_end_time 1177977600** **Note:** *time* must be in UTC format.
- pin_adu struct_valid_01 *n* **- pin_adu struct_valid_02** *n* **- pin_adu struct_valid_03** *n* **- pin_adu struct_valid_04** *n* **- pin_adu dynamic_valid_01** *n*	Use these entries to enable (**1**) or disable (**0**) the predefined validations. For example: **- pin_adu struct_valid_01 1** **- pin_adu dynamic_valid_01 1**
- pin_mta logfile *file_name*	Set *file_name* to the ADU log file. All the debug and error messages generated by ADU are logged to this file. **- pin_mta logfile** *BRM_home*/**sys/diagnostics/pin_adu_validate/adu.pinlog**
- pin_mta loglevel *n*	Set *n* to the level of information to log in the ADU log file. Log level values are as follows: **0**: no logging **1:** log error messages only **2**: log error messages and warnings only **3**: log error messages, warnings, and debug messages **- pin_mta loglevel 2**

Part VIII
Using Self-Care Manager

This part explains how to register and manage customers using Oracle Communications Billing and Revenue Management (BRM) Self-Care Manager. It contains the following chapters:

- Setting Up Customer Self Care with Self-Care Manager

31

Setting Up Customer Self Care with Self-Care Manager

Learn how to set up Oracle Communications Billing and Revenue Management (BRM) Self-Care Manager to display Web pages that your customers can use to access their accounts. It includes a procedure for configuring the application server.

Topics in this document:

To add or change Self-Care Manager functionality, see "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

About Self-Care Manager

Self-Care Manager includes the following components:

- A set of Java Server Pages (JSPs) that display HTML forms in a standard Web browser. The JSPs can display data, such as a list of available packages, from the BRM database.

  You can customize the JSPs to change the appearance of the Web pages or to change the data that you receive from or display to customers. For more information about JSPs, see Sun Microsystems' Java documentation.

- A set of BRM Business Application SDK (BAS) beans that provides an interface between the JSPs and BRM. The BAS beans connect to a Connection Manager (CM) and transfer data to and from BRM.

  A programmer can create custom controllers and other Java components to customize the data that you display or receive from customers. For more information, see "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

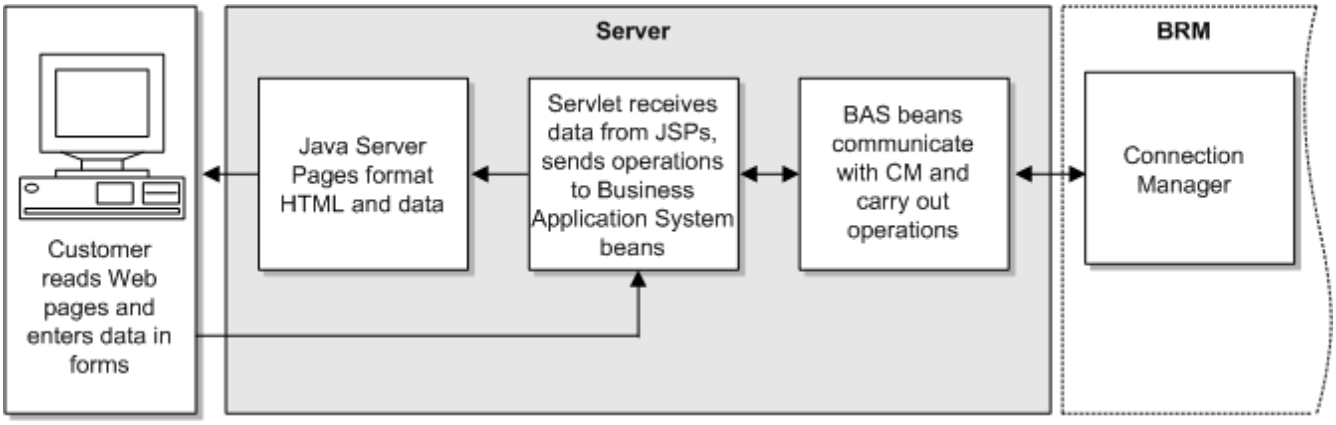
- An application server that processes requests originating from customer Web browsers, manages customer sessions, and transfers data to and from the BAS beans. For more information, see the JavaSoft documentation.

The JSPs and the application server are run by your Web server.

Supported Application Servers

Self-Care Manager supports the Tomcat and WebLogic application servers. Figure 31-1 shows the flow of information.

Figure 31-1 Self-Care Manager System Architecture



For installation instructions, see "Installing Self-Care Manager" in *BRM Installation Guide*.

About Customizing Self-Care Manager

Self-Care Manager can be customized as follows:

- You can change the appearance of Self-Care Manager pages by editing the HTML files, JSPs, and style sheet. This requires HTML knowledge. See "Changing Web Pages" for more information.
- You can modify or add source code to collect additional information from your customers or to provide your customers with additional options. This requires the Customer Center SDK as well as Java programming knowledge.

About Currencies

All accounts created in Self-Care Manager use the default account currency for your BRM system. Your customers do not have the option of selecting a currency when they create an account through Self-Care Manager.

You can change the default currency Self-Care Manager uses by changing the default account currency. See "Setting the Default Account Currency".

If you want customers to have the option of selecting a currency, a programmer can customize Self-Care Manager to add a currency field. See "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

Configuring the Application Server

You can deploy your Self-Care Manager application with these application servers:

- Apache Tomcat. See "Setting Up Apache Tomcat".
- WebLogic. See "Setting Up Oracle WebLogic".

Setting Up Apache Tomcat

This section explains how to deploy your Self-Care Manager application with Apache Tomcat.

> **Note:**
>
> Self-Care Manager is not compatible with Tomcat in a clustered environment. Self-Care Manager should not have the <distributable /> element set in the **web.xml** file.

Requirements

- Your Self-Care Manager **WAR** file. See "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.
- The directory in which Self-Care Manager is installed. The default is *BRM_home***WebKit****WebKit**, where *BRM_home* is the directory in which the BRM server software is installed.

Deploying Your Application with Apache Tomcat

To deploy your Self-Care Manager application with Apache Tomcat:

1. Install Apache Tomcat. For information, see the Apache Tomcat documentation.
2. Deploy your **WAR** file as a new Web component by using the Apache Tomcat interface.
3. Copy the **WebKit.properties** and **Infranet.properties** files from *BRM_home***WebKit****WebKit** to *Tomcat_home***lib**.
4. Stop and restart the Tomcat server.

   Your Self-Care Manager application is now deployed.

Setting Up Oracle WebLogic

This section explains how to deploy your Self-Care Manager application with Oracle WebLogic.

Requirements

- Creating a new WebLogic domain.
- Your Self-Care Manager **WAR** file. See "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.
- *The default* directory in which Self-Care Manager is installed. The default is **C:\Program Files\Portal Software\Self Care Manager**.

Deploying Your Application with Oracle WebLogic

To deploy your Self-Care Manager application with WebLogic:

1. Install WebLogic. For information, see the WebLogic documentation.
2. Create WebLogic domain. For information, see the WebLogic documentation.

3. Copy the **WebKit.properties** and **Infranet.properties** files from **C:\Program Files\Portal Software\Self Care Manager** to *WebLogic_home***mydomain\applications\webkit_en\WEB-INF\classes**.
4. Start the WebLogic server.
5. Deploy your **WAR** file as a new Web component by using the WebLogic interface. For information, see the WebLogic documentation.
6. Stop and restart the WebLogic server.

   Your Self-Care Manager application is now deployed.

Deploying Self-Care Manager

Your Self-Care Manager installation includes the user interface files (JSPs, HTML files, and GIFs) for the standard browser version of Self-Care Manager. The files are in the **htmlui_en** directory.

> **Note:**
>
> This is the directory name for the English locale. Directory names for localized versions use the appropriate locale in place of **en**.

One approach to deploying Self-Care Manager so your subscribers can access it:

- Configure your Web server to look for **index.html** as the default HTML home page.
- Have subscribers enter **http://***hostname***/WebKit/htmlui_en in the Web browser**.

> **Note:**
>
> If your Web server does not support specifying a default home page for each application, subscribers will need to enter the full URL, including the file name for the home page. For example, **http://***hostname***/WebKit/htmlui_en/index.html**.

Supporting Localized Versions of Self-Care Manager

If you have customers in different countries, you can set up localized versions of the Self-Care Manager pages. You obtain localized versions of Self-Care Manager from BRM or use the Localization SDK to create localized versions. In both cases, you install a Self-Care Manager **WAR** file just as you do for English.

You can deploy multiple localized versions of Self-Care Manager in one of these ways:

- **Set up a separate application server for each locale.** You set up multiple application servers with a single Web server. For information, see your application server documentation.

  You can then follow the standard procedure for configuring your application server for each combination of server and locale. For each server, deploy the **WAR** file for a different locale.
- **Set up a separate Web application in your application server for each locale.**

> **Note:**
>
> If you use this method, you will encounter problems later if you try to use opcode and flist logging for troubleshooting.

For information on creating localized versions of Self-Care Manager, see "Localizing Self-Care Manager" in *BRM Developer's Guide*.

Using Self-Care Manager

This section describes how to set up Self-Care Manager for use.

Setting Connection Parameters

Self-Care Manager uses information in the **WebKit.properties** file to establish a connection to BRM. This properties file includes the standard BRM connection parameters, such as database number and login type.

> **Note:**
>
> BRM connection parameters are also in the **Infranet.properties** file.

To set connection parameters:

1. Open the Self-Care Manager properties file (*SelfCareManager_install_dir*/**WebKit.properties**).
2. Edit these entries:

   ```
   user=root.0.0.0.1
   password=password
   host=hostname
   port=11960
   ```

3. Save and close the file.
4. Open the **Infranet.properties** file (*SelfCareManager_install_dir*/**Infranet.properties**).
5. Edit the **infranet.connection** entry; for example:

   ```
   infranet.connection=pcp://root.0.0.0.1:password@host:40010/service/
   pcm_client 1
   ```

   In this example:

   - The login name is **root.0.0.0.1**.
   - The hostname is **host**.
   - The port is **40010**.

6. Save and close the file.

> **Note:**
>
> To improve security, change the default root login name (**root.0.0.0.1**). See "Managing Login Names and Passwords for BRM Access" in *BRM System Administrator's Guide*.

Specifying Which Package List to Display

By default, Self-Care Manager displays packages included in the **webclient** package list. If the **webclient** package list does not exist, Self-Care Manager displays packages included in the **default-new** package list.

To specify a package list for Self-Care Manager:

1. Open the Self-Care Manager properties file (*SelfCareManager_install_dir*/**WebKit.properties**).
2. Enter the name of the package list in the **Infranet.PricingPlan** entry; for example:

   ```
   pricingplan=webclient
   ```

3. Save the file.

Setting the Timeout

You can specify the length of time before a timeout when there is no customer activity.

You change the timeout setting in your application server.

Enabling a Single Login for Multiple Services

By default, Self-Care Manager requires your customers to enter a separate login and password for each service when they purchase bundles with more than one service.

You can set up Self-Care Manager to require only a single login and password that applies to all services. To do this, edit the Self-Care Manager properties file as follows:

1. Open the Self-Care Manager properties file (*SelfCareManager_install_dir*/**WebKit.properties**).
2. Change the **singleLogin** entry to:

   ```
   singleLogin=true
   ```

3. Save and close the file.

Optimizing Self-Care Manager Connection Pool Performance

Self-Care Manager uses connection pooling. When a BRM connection is required for an opcode call, the connection is made for a brief period of time, instead of for the entire duration of a user session, and is then released back to the connection pool. This enables each BRM connection to handle multiple customer connections.

To optimize Self-Care Manager performance, you can change the following connection pooling parameters in the Self-Care Manager properties file (*SelfCareManager_install_dir*/**WebKit.properties**):

- **infranet.bas.connectionpool.size**: The maximum number of connections in the connection pool. More connections can improve performance but put a heavier load on the BRM server. Any value greater than **0** is valid. The default is **4**.

  ```
  infranet.bas.connectionpool.size = 4
  ```

- **infranet.bas.connectionpool.timeout**: The maximum time in milliseconds that customers wait for a BRM connection before getting notified that a connection is not available. If customers do not get a connection within this time, they get an “error communicating with BRM" message.

  You must add this parameter to the Self-Care Manager properties file (**WebKit.properties**) using a value appropriate for your installation. Any value greater than **0** is valid.

  ```
  infranet.bas.connectionpool.timeout = connection_timeout_in_milliseconds
  ```

  The default is **30000** (30 seconds).

Adding a Service to the Self-Care Manager Home Page

The Self-Care Manager home page includes a list of services available to your customers. If you add an optional manager to your BRM system, and that optional manager uses Self-Care Manager, you need to add those service types to the list in the home page.

> **Note:**
>
> If you have multiple Self-Care Manager home pages, you need to add those service types to each home page.

To add a service:

1. Open the Self-Care Manager home page with an HTML editor or text editor. The default Self-Care Manager home page is **index.html**.
2. Locate this section of the file:

   ```
   <SELECT Name="service" Size="1">
        <OPTION Value="ip" SELECTED>ip
        <OPTION Value="email">email
        <OPTION Value="telephony">telephony
    </SELECT>
   ```

3. For each service you want to appear on the list, add a new option to this list. For **Value**, enter the BRM name of the service. The name outside the **OPTION** tag is the name displayed on the Web page.

   For example:

   ```
   <OPTION Value="gsm/sms">GSM SMS
   <OPTION Value="gsm/telephony">GSM Telephony
   <OPTION Value="gsm/data">GSM Data
   ```

4. Save and close the file.

Troubleshooting Self-Care Manager

This section contains troubleshooting information about these Self-Care Manager issues:

- The error logs show an “error communicating with Portal" message

- The Web pages display errors

The error logs show an “error communicating with Portal" message

To debug this type of error, you turn on opcode and flist logging for the Java PCM. This logs the input and output flists for each opcode that Self-Care Manager calls. You can use these flists to see problems communicating with BRM.

To turn on opcode and flist logging:

1. Open the **Infranet.properties** file in a text editor. The file is located in your Self-Care Manager installation directory.
2. Add these lines to the file:

   ```
   infranet.log.opcodes.enabled=true
   infranet.log.opcodes.file=pathname/opcodes.log
   ```

   where *pathname is* the path to the directory for the log file. For example:

   ```
   infranet.log.opcodes.file=d:/temp/opcodes.log
   ```

   > **Note:**
   >
   > You must use forward slashes in the directory paths.

3. Stop and restart the application server.
4. Save and close the file.
5. Reproduce the error and check the file *applicationserver_install_dir***/servers/default/javaPCM.log**.

The Web pages display errors

The Web pages might display an error message or display “Click to continue" in place of the correct page. To debug this type of problem, you enable debugging in the **error.jsp** file so that exceptions are written to the log file **debug_errorlog.jsp**. This prints a stack trace, which can help developers locate where an exception occurred.

To enable debugging in the **error.jsp** file:

1. Open the appropriate version of the **error.jsp**, located in the **htmlui_en** directory in the Self-Care Manager installation directory.
2. Uncomment this line by removing <% and %>:

   ```
   <%@ include file = "debug_errorlog.jsp" %>
   ```

3. Save and close the file.

Using the Default HTML Web Pages

Self-Care Manager supplies a set of JSPs that you can use as part of your customer self-care website. These JSPs serve HTML pages.

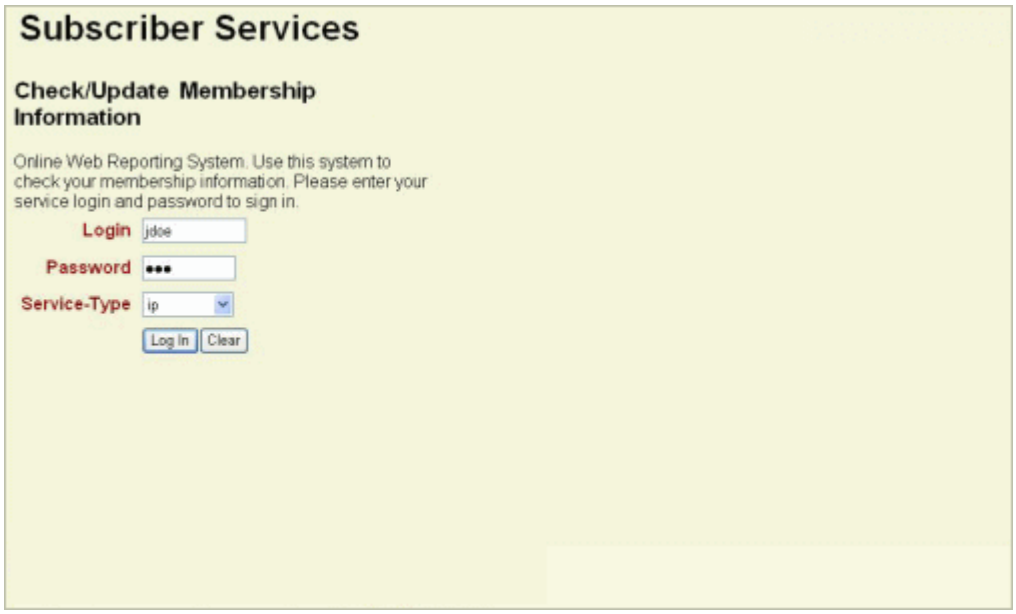
This section describes Self-Care Manager's default HTML pages and provides an overview of the functionality they offer.

For information on customizing these pages, see "Changing Web Pages" and "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

Logging In

Figure 31-2 shows the login page your customers see when they access your self-care website.

Figure 31-2 Login Page for Self-Care Website



Self-Care Manager supports email, IP, and telephony service types.

Accessing Account Information

After customers log in to your website on the login page, the Account Information page appears. This page displays the details of the account bill unit (**/billinfo** object) and balance groups associated with the bill unit, and the current balance of all the balance groups.

To view the details of the other bill units in the account, customers click the **Other Bill Units Information** link on the Account Information page and select the bill unit. The details will be displayed in the Account Information page. For information, see "Accessing Account Information".

From this page, customers can perform the following tasks:

- Finding or Searching for and Viewing Events
- Applying Voucher Top-Ups
- Viewing Resource Reservation Details
- Viewing Invoices
- Viewing Account Activity
- Viewing Products or Offers Purchased

- Paying Bills Online
- Viewing Service Details for a Bill Unit
- Viewing Bill Units in an Account
- Changing Login Name, Password, or Account Status

Finding or Searching for and Viewing Events

Customers can specify the search criteria and view the events of the selected bill unit. To enable this feature, you must configure the **item types** in the **WebKit.properties** file.

To specify the search criteria, customers click the **Event Search** link on the Account Information page. The Event Search page appears. Customers then specify the search criteria and click **Search**.

To view the events of other bill units in the account, see "Viewing Bill Units in an Account".

Based on the search criteria and the **item types** configured in **WebKit.properties**, events are retrieved and displayed in the Event Search page.

For example:

- To view all the events related to *cycle_forward*, add */cycle_forward* to the key *EventSearch.ItemTypes* in **WebKit.properties:**

  ```
  EventSearch.ItemTypes=/cycle_forward
  ```

- To view all the events related to *adjustment and payment*, add */adjustment* and */payment* to the key *EventSearch.ItemTypes* in **WebKit.properties:**

  ```
  EventSearch.ItemTypes=/adjustment,/payment
  ```

Applying Voucher Top-Ups

Customers use vouchers to top-up their currency and noncurrency account balances. They can top-up one currency balance and an unlimited number of noncurrency balances. For more information, see "About Standard Top-Ups" in *BRM Configuring and Collecting Payments*.

To top-up their account with a voucher, customers click the **Voucher Top-Up** link on the Account Information page. The Voucher Top-Up page appears. Customers can then enter the voucher ID and pin and click **Validate** to check the balances and validity of the specified voucher. If the voucher is valid, the **Apply** button is enabled.

If there are multiple balances in the bill unit, customers can select the balance groups whose account balance they want to top up.

To apply the vouchers immediately and top-up the account balances, customers select the required balance groups, select **Allocate Now**, and click **Apply**.

Customers can also choose just to apply the voucher to the balance groups and not top up the account balance. To do this, the customer clicks **Apply** without selecting **Allocate Now**. The voucher can later be allocated but *only* through Customer Center.

Viewing Resource Reservation Details

Customers can view the resource reservation details of a bill unit. To do this, customers click the **Resource Reservation** link on the Account Information page. It displays available credit limit, total bill amount, amount reserved, and available balance of all the balance groups in a bill unit.

To view the reservation details, customers click the **Reservations** link and the details are displayed in the Resource Reservations Details page.

If the credit limit of any of the balance groups is unlimited, the **Credit Limit** value is displayed as **Unlimited**.

To find the balance group, which has unlimited credit balance customers, click **the Unlimited** link displayed next to the **Credit Limit**. The Included Services page appears. See "Viewing Service Details for a Bill Unit".

Viewing Invoices

Customers can view their invoice details by generating a report of their past bills. To view the invoice details, customers click the **Invoice** link on the Account Information page. The Invoice Selection page.

Customers can then select new start and end dates for the report or keep the default dates that represent the last billing cycle. After clicking **Submit**, the Invoice page appears, which lists invoices available for the selected period. In this page, customers click the invoice they want to view.

Viewing Account Activity

To view their account details, customers can generate a report. To generate a report, customers click the **Account Activity** link on the Account Information page.

Customers can then select new start and end dates for the report or keep the default dates which represent the last billing cycle.

After clicking **Submit**, the Account Activity page appears which shows the account's details; for example, customer's name, address, and event details in chronological order.

If the customer uses additional services, some types of usage activity appear on separate pages. Links for the additional pages appear on the line just above the list of events.

Viewing Products or Offers Purchased

To view the products or offers purchased, customers click the **Products** link on the Account Information page. The Product Information page.

If a discount was purchased as part of a bundle, an icon appears before the bundle name.

Paying Bills Online

To pay bills online, customers click the **Online Payment** link on the Account Information page.

To pay the bill, customers enter an amount and credit card information and click **Submit**.

BRM updates the customer's balance the next time you run billing and collects the BRM-initiated payment.

For more information, see "About BRM-Initiated Payment Processing" in *BRM Concepts*.

To view online bill payment records, customers click the **Online Payment Audit** link on the Online Payment page. A page appears on which they enter start and end dates. Self-Care Manager displays the payments made between the start and end dates.

Viewing Service Details for a Bill Unit

To view the list of services and the details associated with all the balance groups in a bill unit, customers click the **Included Services** link on the Account Information page. The Service Details page appears and displays the details of the services.

Viewing Bill Units in an Account

To view all the bill units in an account, customers click the **Other Bill Units Information** link on the Account Information page. The Service Level Bill Unit Information page appears and displays all the bill units.

To view the details of a specific bill unit, click on the bill unit. The details are displayed in the Account Information page. For information, see "Accessing Account Information".

Changing Login Name, Password, or Account Status

Customers can change their login name, password, or account status by clicking the appropriate link on the Account Information page.

If customers click **Change a Login** or **Change Status**, the Change Service Login page appears.

The account status can be active, inactive, or closed.

Changing Web Pages

This section describes editing Self-Care Manager files to change the appearance of the pages and to make other modifications. You can make these types of changes if you are familiar with HTML tags and JSP tags. For information on making more extensive customizations to Self-Care Manager, see "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

Editing Self-Care Manager Files

To change the appearance of Web pages, edit the following user interface files, located by default in the **htmlui_en** directory:

- **JSPs**: The files containing the text and forms that provide the user interface on the pages. JSPs are HTML pages with added JSP tags. You can use any text editor and some HTML editors to edit the JSPs.

  For information on modifying these pages, see "Editing JSPs".

- **HTML pages**: The HTML account creation page and files with common HTML code used by the JSPs. For example, the **header.html** file includes the HTML **HEAD** tag. You can use a text editor or HTML editor to edit these files.

- **Cascading style sheet (CSS)**: The file that defines the appearance of the HTML pages. Edit this file to make global formatting changes, including fonts, text alignment, and margins. You can use a text editor or a CSS editor to edit this file.

- **GIF files**: The graphics displayed in the HTML pages. You can edit the files with a graphics program or replace them with your own graphics.

> **Note:**
>
> Before editing, make a copy of the default pages, including all HTML pages and JSPs, and edit and test the copies.

When you edit a Self-Care Manager file, you should edit the version located in the Self-Care Manager **WAR** file. This ensures that the customized files are used if you reconfigure your servlet engine for Self-Care Manager in the future. See "Modifying the Self-Care Manager WAR File".

You can also use the Customer Center SDK to modify Self-Care Manager files from the **WAR** file. See *BRM Developer's Guide*.

Editing JSPs

JSPs use standard HTML formatting. In addition, the JSPs use special code to display information, or to display a text entry field for customer input. Each JSP tag starts with **<%** and ends with **%>**.

You can move data input and display codes from one location on the page to another. You can also delete them. For example, if you do not offer the IP telephony service, you can delete all code that displays IP telephony information.

> **Note:**
>
> - When editing JSPs, be careful to maintain the syntax when moving or deleting code. For example, a missing %> from a JSP tag will cause problems.
> - When removing code, do not remove input entry points for information that is required for account creation. See "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.
> - Do not copy data input and display codes from one page to another. Each code works only on the page that includes it.

For more information about editing JSPs, see Sun Microsystems' JSP documentation or other JSP reference documents. For more extensive JSP customization, see "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

Modifying the Self-Care Manager WAR File

When you edit Self-Care Manager files, you should edit the versions in the Self-Care Manager **Web Application Archive** (WAR) file. This file is created when you install Self-Care Manager, and it is the source for the Self-Care Manager files that the servlet engine copies when you configure it to run Self-Care Manager.

If you configured your servlet engine before modifying individual JSPs or other files, you also must copy your modified files to the servlet engine directory where Self-Care Manager files are located.

By modifying the files in the **WAR** file, you make sure that you do not overwrite your modified files if you reconfigure your servlet engine for Self-Care Manager in the future.

Following are instructions for modifying the Self-Care Manager **WAR** file. Alternatively, you can use the Customer Center SDK to create a customized **WAR** file. See *BRM Developer's Guide*.

To modify the Self-Care Manager **WAR** file, you need to install the Java Development Kit (JDK) on your system. The system running your application server already has the JDK. For information on supported versions, see "BRM Software Compatibility" in *BRM Compatibility Matrix*.

To modify the Self-Care Manager **WAR** file:

1. Copy the *SelfCareManager_install_dir*/**WebKit/webkit_en.war** file to a temporary directory.
2. Extract the files from the **WAR** file using the **jar** command, as follows:

   ```
   jar xvf webkit_en.war
   ```

3. Delete or move the copy of **webkit_en.war** in the temporary directory.
4. Edit any files in the **htmlui_en** directory.
5. Create a new **WAR** file using the **jar** command from the temporary directory where you extracted the original files:

   ```
   jar cvf webkit_en.war .
   ```

6. Copy this **WAR** file to *SelfCareManager_install_dir*/**WebKit**.

When you set up Self-Care Manager as an application in your servlet engine, it will include the files you modified.

If you have already set up Self-Care Manager in the servlet engine, copy the modified files to the correct location in the servlet engine's directory.

Files Used by Self-Care Manager

This section describes the various types of files used by Self-Care Manager.

JSPs Used by Self-Care Manager

Table 31-1 describes the JSPs used by Self-Care Manager:

> **Note:**
>
> These pages use the Business Application SDK (BAS) component collection (**PIAComponentCollection**) to hold BAS lightweight components (that is, **PLightComponentHelper**).

Table 31-1 Self-Care Manager JSPs

File name	Description
change_login_form.jsp	Enables customers to change the login name for a service.
change_passwd_form.jsp	Enables customers to change a password.
change_status_form.jsp	Enables customers to change account status.
constants.jsp	Defines strings for constants commonly used in other Self-Care Manager files.

Table 31-1 (Cont.) Self-Care Manager JSPs

File name	Description
debug_errorlog.jsp	Enables error logging for debugging.
error.jsp	Handles errors and displays error messages for all JSPs.
footer.jsp	Contains standard information displayed at the end of all JSPs.
invoice_selection.jsp	Enables customers to select the start time and end time used for generating the invoice.
load_session.jsp	Used by other JSPs to retrieve commonly used account data saved during the session, such as the name and address for the account.
login_verify.jsp	Verifies the login name and password and displays confirmation when a customer logs in.
logout.jsp	Displays confirmation that a customer logged out and ends the session.
online_payment.jsp	Enables customers to pay an outstanding bill online with a credit card.
online_payment_audit.jsp	Enables customers to search for records of online bill payments. The customer specifies a start date and end date.
purchase_plan_form.jsp	Enables customers to purchase additional packages.
selection.jsp	Enables customers to select the day, month, and year when specifying a time range for viewing account activity or invoices.
session_id.jsp	Saves the current user's session ID.
show_invoice.jsp	Displays a customer invoice.
usage_report.jsp	Displays details about customer activity, such as a list of IP telephony calls. HTML pages use **UsageReportGeneral.jsp** instead.
usage_selection.jsp	Enables customers to set the start and end dates for a usage report. For example, a customer can specify to see IP telephony calls for more than a single month.
UsageReportCommon.jsp	Included in the other usage report JSPs to do event search tasks that are common to all the usage report JSPs.
UsageReportContent.jsp	Displays details about a customer's content usage.
UsageReportGeneral.jsp	Displays details about general customer activity, including cycle forward fees, purchase fees, adjustments, and IP and email usage. Other usage report JSPs display activity for specific types of usage.
UsageReportGprs.jsp	Displays details about a customer's mobile data (GPRS) usage.
UsageReportGsm.jsp	Displays details about a customer's mobile voice (GSM) usage.
UsageReportSms.jsp	Displays details about a customer's mobile messaging (SMS) usage.
UsageReportView.jsp	Included in the other usage report JSPs to display the data from a usage report on the Web page.
view_balance.jsp	Displays the customer's account balance summary and has links to resource or balance reservation, event search, and voucher top-up pages.
view_invoice.jsp	Enables customers to specify which invoice to display.
view_product.jsp	Displays information about a customer's purchased products or charge offers.
AccountBalancePrepaid.jsp **AccountBalancePrepaidView.jsp** **ReservationInfo.jsp** **ReservationInfoView.jsp**	Display the resource or balance reservation details of a bill unit.

Table 31-1 (Cont.) Self-Care Manager JSPs

File name	Description
Event_details.jsp **Event_search.jsp** **eventsearch_dateselection.jsp**	Enable customers to view the events of the selected bill unit.
Voucher_details.jsp	Enables customers to top-up account balances by using a voucher.

HTML Pages Used by Self-Care Manager

Table 31-2 shows the HTML files used by Self-Care Manager:

Table 31-2 HTML Files Used by Self-Care Manager

File name	Description
footer.html	HTML code that you can use as a footer on every JSP page by including this file name in the JSP. For example: <jsp:include page="footer.html"/>
header.html	HTML code that you can use as a header on every JSP page by including the file name in the JSP.
index.html	The default account creation home page. Customers see this page first when they go to your website.

Other Files Used by Self-Care Manager

Self-Care Manager also uses these files:

- **Style sheet**: The HTML version of the JSPs use the style sheet **infranet_general.css**.
- **GIF files**: The HTML version of the JSPs use several GIF files. Most are stored in **htmlui_en/graphics**, others are stored in **htmlui_en**.

Part IX

Customer Management Utilities

This part provides information about the customer management utilities provided with Oracle Communications Billing and Revenue Management (BRM). It contains the following chapters:

- Customer Management Utilities

32

Customer Management Utilities

Learn about the Oracle Communications Billing and Revenue Management (BRM) customer management utilities.

Topics in this document:

load_config_business_event

Use this utility to load or retrieve the list of business events that can be associated with notification specifications. This list is stored in the **/config/business_events** object and displayed in the Billing Care Trigger Criteria screen.

For more information, see "Configuring Billing Care to Display Business Events".

> **Note:**
>
> To connect to the BRM database, this utility needs a configuration (**pin.conf**) file in the directory from which it is run. For information about creating configuration files for BRM utilities, see "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_config_business_event [-n filename] [-w filename] [-p filename] [-o filename] [-r]
[-d] [-h] [-v]
```

Parameters

-n *filename*
Creates a new **/config/business_events** object by loading the contents of the XML file into the database schema.

-w *filename*
Writes information from the **/config/business_events** object into the specified XML file.

-p *filename*
Loads business events from the **payload_config** file in XML format. If this option is not provided, it loads from the **business_events** file format.

-o *filename*
Loads business events from the specified XML file and overwrites entries in the **/config/business_events** object. Duplicates are ignored.

-r
Deletes all instances of the **/config/business_events** object from the BRM database.

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the configuration objects have not been loaded into the database.

-h
Displays the syntax and parameters for this utility.

-v
Displays information about successful or failed processing as the utility runs.

Results

If the **load_config_business_event** utility does not notify you that it was successful, look in the utility log file (**load_config_business_event.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

load_pin_business_profile

Use this utility to load business profiles and validation templates into the BRM database.

See the following topics:

- About Business Profiles
- Setting Up Business Profiles and Validation Templates
- Editing the Business Profile Configuration File

Location

*BRM_home***/bin**

Syntax

```
load_pin_business_profile [-d] [-h] [-r] [-t] [-v] filename
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility seemed to run without error but the data was not loaded into the database.

-h
Displays syntax and parameters for this utility.

-r
Retrieves the business profile and validation template data from the BRM database and saves it in an XML file.

-t
Runs the utility in test mode to validate the XML file against its schema definition. This option does *not* create, modify, or delete any business profile or validation template objects in your BRM database.

> **Note:**
>
> To avoid load errors based on XML content problems, run the utility with this option *before* loading data into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_business_profile** *other_parameters* **–v >** *filename***.log**

filename
The name and location of the business profile configuration file. The default file is *BRM_home***/sys/data/config/pin_business_profile.xml**, but the utility can take any XML file name as a parameter as long as the file's contents conform to the appropriate schema definition.
If you copy filename to the same directory from which you run the load utility, specify only the file name. If you run the command in a different directory from where filename is located, you must include the entire path for the file.
In addition, filename must be in the same directory as the default *BRM_home***/sys/data/config/business_configuration.xsd** file.

Results

This utility notifies you only if it encounters errors. Look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

> **Note:**
>
> You must stop and restart the Connection Manager (CM) to make new business profile and validation template data available.

load_pin_customer_segment

Use this utility to load your customer segment definitions into the BRM database.

For information about customer segments, see "Creating and Managing Customer Segments".

Location

*BRM_home***/bin**

Syntax

```
load_pin_customer_segment [-t] [-v] [-d] [-h] pin_customer_segment_file
```

Parameters

-t
Runs the utility in test mode to validate the **XML** file format against the **XSD** file, and does not load the data or overwrite any existing data. Use this option before loading data into the **/config** object.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_customer_segment** *other_parameters* **–v >** *filename***.log**

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-h
Displays help information for using this utility.

pin_customer_segment_file
The name and location of the **XML** file that stores localized strings for customer segments. The **XML** file is referenced by the **pin_business_configuration.xsd** file; therefore these files must be located in the same directory. The default customer segment file is in *BRM_home***/sys/data/config**.
If you copy the **pin_customer_segment.xml** file and the **pin_business_configuration.xsd** file to the same directory from which you run the **load_pin_customer_segment** utility, you do not have to specify either the path or the file name.
If you run the command in a different directory from where the **pin_customer_segment.xml** file is located, you must include the entire path for the file.

Results

The **load_pin_customer_segment** utility notifies you when it successfully creates the **/config/customer_segment** object.

If the **load_pin_customer_segment** utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

> **Note:**
>
> You must restart the Connection Manager to start recording your customer login failure preferences.

load_pin_notify

Use this utility to add an event notification list to your BRM database. See "Loading the Event Notification List" in *BRM Developer's Guide*.

For general information about event notification, see "Using Event Notification" in *BRM Developer's Guide*.

Location

*BRM_home***/bin**

Syntax

```
load_pin_notify [-d] [-v] [-h] pin_notify_file
```

Parameters

-d
Specifies debug mode. The utility logs messages to the **default.pinlog** file in the current directory or in a directory specified in the utility configuration file. Use this parameter for troubleshooting when the utility runs with no errors but the notification file is not loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
> **load_pin_notify** *other_parameters* **–v >** *filename***.log**

-h
Displays the syntax and parameters for this utility.

pin_notify_file
The name and location of the configuration file that contains the event notification list you want to load into your database. The default event notification configuration file, **pin_notify**, is in *BRM_home***/sys/data/config**.
For more information, see "Merging Event Notification Lists" in *BRM Developer's Guide*.

> **Note:**
>
> If you copy the *pin_notify_file* to the same directory from which you run **load_pin_notify**, you do not have to specify the path or the file name.

Results

This utility notifies you when it successfully creates the **/config/notify** object.

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

> **Note:**
>
> You must restart the Connection Manager to make new events and the corresponding opcodes available.

load_pin_verify

Use this utility to load your preferences for recording customer login failures into the BRM database. See "Recording Login Failures".

Location

*BRM_home***/bin**

Syntax

```
load_pin_verify pin_verify_file
```

Parameters

pin_verify_file
The name and location of the file that contains preferences for recording login failures. The default **pin_verify** file is in *BRM_home***/sys/data/config**.

> **Note:**
>
> If you copy the edited **pin_verify** file to the same directory from which you run the **load_pin_verify** utility, you do not have to specify the path or the file name.

Results

The **load_pin_verify** utility notifies you when it successfully creates the **/config/verify** object.

If the **load_pin_verify** utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

> **Note:**
>
> You must restart the Connection Manager to start recording your customer login failure preferences.

load_transition_type

Use this utility to load custom bundle and package transition types into the BRM database.

For information, see "Creating Custom Transition Types for Bundles and Packages".

Location

*BRM_home***/bin**

Syntax

```
load_transition_type [-v] [-d] [TransitionTypeFile]
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

TransitionTypeFile
The name and location of the file that contains your custom transition types. By default, the utility uses **pin_transition_type**.

If you run the command in a different directory from where the **pin_transition_type** file is located, you must include the entire path for the file.

Results

This utility notifies you only if it encounters errors. Look in the **default.pinlog** file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.

pin_contracts

Use this utility to renew customer contracts or cancel contracts that have expired.

> **Note:**
>
> To connect to the BRM database, this utility needs a configuration (**pin.conf**) file in the directory from which it is run. For information about creating configuration files for BRM utilities, see "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

BRM_home/**bin**

Syntax

```
pin_contracts [-renew | -expire | -expire-renewed] [-help] [-verbose] [-test]
```

Parameters

-renew
Renews all contracts that expire today and have auto-renewal enabled.

-expire
Cancels all non-renewable contracts that expire today.

-expire-renewed
Cancels all contracts that have already been renewed and are now at the end of their second commitment period.

-help
Displays the syntax and parameters for this utility.

-verbose
Displays information about successful or failed processing as the utility runs.

-test
Tests the utility, but does not affect accounts or contracts. Use this parameter to see which contracts would be renewed or canceled.

Results

If the **pin_contracts** utility does not notify you that it was successful, look in the utility log file (**pin_contracts.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_deferred_act

Use this utility as part of your daily billing to run deferred actions. For example, if a CSR has scheduled an account to become inactive, the **pin_deferred_act** utility performs the status change on the scheduled date. This utility is included in the **pin_bill_day** script.

See "Scheduling Status Changes in Advance".

Location

*BRM_home***/bin**

Syntax

```
pin_deferred_act [-report|-purge|-retry] [-deferred_notifications]
                 [-opcode opcode_name] [-status pending|done|error]
                 [-start mm/dd/yy] [-end mm/dd/yy] [-verbose] [-test] [-help]
```

Parameters

-report
Displays the progress and current state of **/schedule** objects.
This parameter can take one or more of these options as search criteria to filter the results of your report:

- **-opcode** *[opcode_name]*
- **-status pending|done|error**
- **-start** *mm/dd/yy* or yyyy
- -**end** *mm/dd/yy* or yyyy

For example:

```
pin_deferred_act -report -start 01/10/03 -end 01/24/03 -verbose
```

-purge
Purges from the Oracle Communications Billing and Revenue Management (BRM) database all **/schedule** objects whose actions have been run successfully. This helps reduce the size of your database.
This parameter can take one or more of these options as search criteria for purging:

- **-opcode** *[opcode_name]*
- **-status pending|done|error**
- **-start** *mm/dd/yy* or yyyy
- -**end** *mm/dd/yy* or yyyy

For example:

```
pin_deferred_act -purge -start 01/10/03 -end 01/24/03 -verbose -opcode
PCM_OP_BILL_MAKE_BILL_NOW
```

-retry
Retries all the **/schedule** objects whose schedule actions have failed to run and whose status is marked as ERROR.
This parameter can take one or more of these options as search criteria for purging:

- **-opcode** *[opcode_name]*
- **-status pending|done|error**
- **-start** *mm/dd/yy* or yyyy
- -**end** *mm/dd/yy* or yyyy

For example

```
pin_deferred_act -retry -start 01/10/03 -end 01/24/03 -verbose
```

-deferred_notifications
Runs deferred actions against **/schedule/notification** objects only. For information, see "About Events Occurring Outside of Delivery Times".

-opcode *[opcode_name]*
Used as search criteria by the **–report**, **–purge**, and **–retry** parameters for retrieving **/schedule** objects containing the specified opcode responsible for the deferred action. For example:

```
pin_deferred_act -report -start 01/10/03 -end 01/24/03 -verbose -opcode
PCM_OP_BILL_MAKE_BILL_NOW
```

-status pending|done|error
Used as search criteria by the **–report**, **–purge**, and **–retry** parameters for retrieving **/schedule** objects having the specified status. For example:

```
pin_deferred_act -retry -start 01/10/03 -end 01/24/03 -verbose -status ERROR
```

-start *mm/dd/yy* or *yyyy*
-end *mm/dd/yy* or *yyyy*
Used as search criteria by the **–report**, **-purge**, and **–retry** parameters for retrieving **/schedule** objects with an execution date matching the **start** and **end** dates specified. The value you supply for the **start** date is inclusive, but the value for the **end** date is non–inclusive and also defaults to the current date. If a **start** date is not specified, this utility retrieves all valid **/schedule** objects up to the specified **end** date. If an **end** date is not specified, this utility uses the current date as the end date and retrieves all valid **/schedule** objects until the current date.

-verbose | *-v*
Displays information about successful or failed processing as the utility runs.

> **Note:**
>
> This parameter is always used with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename*.log with the name of the log file:
> **pin_deferred_act** any_other_parameter **–v >** *filename***.log**

-test
Runs a test to find out how many accounts meet the criteria without performing the action. The test has no effect on the accounts. This is most useful when run with the **-verbose** option.
-help| -h
Displays the syntax and parameters for this utility.

Results

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

When it is called internally by the **pin_bill_day** script, the **pin_deferred_act** utility logs error information in the **pin_mta.pinlog** file.

pin_deposit_calc_interest

Use this utility to search for appropriate purchased deposits and calculate the interest amount based on the frequency.

The utility looks for the purchased deposits which match the following criteria:

- The balance amount and the deposit amount are greater than zero.
- The last interest calculation date is less than the current date and is not zero.
- The next interest calculation date is greater than or equal to the current date.

> **Note:**
>
> The final interest calculation frequency is calculated based on the interest offset and the offset unit. The offset unit value can be either days, weeks, months, or years. If the effective date is updated, the interest is calculated on daily basis.

Location

BRM_home **/bin**

Syntax

```
pin_deposit_calc_interest [-verbose] [-help]
```

Parameters

-verbose

Displays information about successful or failed processing as the utility runs.

-help

Displays the syntax and parameters for this utility.

Results

If **pin_deposit_calc_interest** does not notify you that it was successful, look in the utility log file (**pin_deposit_calc_interest.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_deposit_release_purchased_deposit

Use this utility to search for purchased deposits for which the end date is less than the current system date, then release the deposit amount.

Location

BRM_home/**bin**

Syntax

```
pin_deposit_release_purchased_deposit [-help] [-verbose] [-test]
```

Parameters

-help

Displays the syntax and parameters for this utility.

-verbose

Displays information about successful or failed processing as the utility runs.

-test

Tests the utility but does not affect the purchased deposit. Use this parameter to check if the deposit amounts are releasing correctly.

Results

If **pin_deposit_release_purchased_deposit** does not notify you that it was successful, look in the utility log file (**pin_deposit_release_purchased_deposit.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_deposit_transfer_deposit

Use this utility to transfer the deposit balance amount from one account or service to another account or service. The input for this utility is a CSV file.

Location

BRM_home/**bin**

Syntax

```
pin_deposit_transfer_deposit -f input_file.csv [-verbose] [-test] [-help]
```

Parameters

-f *input_file*.**csv**

Specifies the name and location of the file that specifies how BRM will transfer the deposit amount. For example, **\deposit\transfer_deposit.csv**.

For more information on the CSV file, see "About Transferring Multiple Deposits Simultaneously".

-verbose

Displays information about successful or failed processing as the utility runs.

-help

Displays the utility's syntax and parameters.

Results

pin_deposit_transfer_deposit handles each deposit transfer as a separate transaction. If any single deposit transfer fails, BRM does not need to roll back the entire deposit transfer. Successful transfers remain in place.

If **pin_deposit_transfer_deposit** does not notify you that it was successful, look in the utility log file, ${PIN_LOG_DIR}**/pin_deposits/pin_deposit_transfer_deposit.pinlog**, to find any errors.

pin_gen_notifications

Use this utility to generate notifications either several days before or several days after the following occurs:

- A customer's balance expires
- A customer's bill is due
- A collections action is performed against a customer
- A customer's installment payment is due
- A customer's subscription expires
- A customer's subscription is due for renewal
- A customer's service lifecycle state changes

The utility determines when to generate notifications for expiring resources by reading the ***/config/notification_spec*** object.

For more information, see "About Generating Notifications In Advance" and "About Generating Notifications After an Event Occurs".

> **Note:**
>
> To connect to the BRM database, this utility needs a configuration (**pin.conf**) file in the directory from which it is run. For information about creating configuration files for BRM utilities, see "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
pin_gen_notifications  -custom BusinessEventName|-svc_lifestate_change|-balance_expiry|-
product_expiry|
                       |-subscription_renewal_due|-bill_due|-collections_action
[action]|-installment due_date|end_date
                       [-after_expiry] [-search_level self|account|bal_grp|service|
billinfo] [-virtual_start_time timestamp]
                       [-start timestamp -end timestamp] [-file filename]
                       [-help] [-verbose] [-test]
```

Parameters

-custom *BusinessEventName*
Generates notification events for any custom business events.

-svc_lifestate_change
Generates the **/event/notification/service/state_change/pre_expiry** notification event for service lifecycle state changes a specified number of days before the event occurs.

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/service/state_change/post_expiry** notification event for service lifecyle state changes a specified number of days ago.

-balance_expiry
Generates the **/event/notification/balance/expiry** notification event for each customer balance whose validity period expires in the specified number of days before the event occurs.

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/balance/post_expiry** notification event for each customer balance whose validity expired the specified number of days ago.

-product_expiry
Generates an **/event/notification/product/expiry** notification event for each customer subscription that expires in the specified number of days.

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/product/post_expiry** notification event for each customer subscription that expired the specified number of days ago.

-subscription_renewal_due
Generates an **/event/notification/subscription/renewal_due** notification event for each customer subscription that renews in the specified number of days.

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/subscription/post_renewal_due** notification event for each customer subscription that was due for renewal a specified number of days ago.

-bill_due
Generates an **/event/notification/billing/due** notification event for each customer bill that is due in the specified number of days.

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/billing/post_due** notification event for each customer bill that expired a specified number of days ago.

-collections_action [*action*]
Generates an **/event/notification/collections_action/due** notification event for each collections action that is almost due. If you run **-collections_action** without the *action* option, the utility generates notification events for *all* types of collections actions.
To generate notification events for just one collections action type, set the *action* option to one of the following:

- **close_billinfo**: Generates notifications before bill units are closed.
- **collect_payment**: Generates notifications before collections payments are due.
- **dunning_letter**: Generates notifications before dunning letters are sent to customers.
- **finance_charge**: Generates notifications before finance charges are applied to accounts.
- **inactive_billinfo**: Generates notifications before bill units are inactivated.
- **invoice_reminder**: Generates notifications before invoice reminders are sent to customers.
- **late_fee**: Generates notifications before late fees are applied to accounts.
- **manual_action**: Generates notifications before collections agents make phone calls to customers.
- **promise_to_pay**: Generates notifications before promise-to-pay payments are due.
- **policy_action**: Generates notifications before policy actions are performed for bill units.
- **refer_to_outside_agency**: Generates notifications before bill units are referred to an outside agencies.
- **writeoff_billinfo**: Generates notifications before bill units are written off.

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/collections_action/post_due** notification event for each collections action that is past due.

-installment due_date | end_date
Generates an **/event/notification/installment_schedule/due** notification event for each installment that is due on the current date (**-installment due_date**) or that ends on the current date (**-installment end_date**).

> **Note:**
>
> If used with the **-after_expiry** parameter, it generates an **/event/notification/installment_schedule/post_due** notification event for each installment action that is past due.

-after_expiry
Generates notifications after an event has occurred. For example, a configured number of days after a customer's subscription has expired.

-virtual_start_time *timestamp*
The utility assumes the passed value as the current system time. The timestamp must be in the format *mm/dd/yyyy-HH:mm:ss*. For example: 04/23/2030-18:22:45.

-search_level self | account | bal_grp | service | billinfo
Consolidates multiple in-advance or post-expiration events that expire at the same time into one notification that is sent to an external notification application. You can consolidate events at different levels:

- **-search_level self**: Creates separate notifications for each business event.
- **-search_level account**: Creates a single notification for an account with multiple events expiring at the same time.
- **-search_level bal_grp**: Creates a single notification for a balance group with multiple events expiring at the same time.
- **-search_level service**: Creates a single notification for a service with multiple events expiring at the same time.
- **-search_level billinfo**: Creates a single notification for a bill unit with multiple events expiring at the same time.

-start *timestamp* -end *timestamp*
Specifies to search for objects that expire during the specified time period. The timestamp must be in the format *mm/dd/yyyy-HH:mm:ss*.
When this option is used, the time constraints in **/config/notification_spec** are ignored.

-file *filename*
Specifies to generate notification events for the objects listed in the specified file only. For information about the file's format and contents, see "Creating a File of POIDs".

-help
Displays the syntax and parameters for this utility.

-verbose
Displays information about successful or failed processing as the utility runs.

-test
Tests the utility by retrieving the list of balance resources that are about to expire, but it does not generate notification events.

Results

If the **pin_gen_notifications** utility does not notify you that it was successful, look in the utility log file (**pin_gen_notifications.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_monitor_balance

Use this utility to update monitored balances and to check whether the balances have crossed thresholds.

For more information on using the **pin_monitor_balance** utility, see "Running pin_monitor_balance to Update Monitored Balances".

> **Note:**
>
> To connect to the BRM database, this utility needs a configuration file in the directory from which you run it. The **pin.conf** file for this utility is in *BRM_home***/apps/pin_monitor**. See "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
pin_monitor_balance [-d] [-v] [-t] [-h]
```

Parameters

-d
Prints error logs for debugging.

-v
Displays detailed information on status and error messages as the utility updates balances and generates notification events.

-t
Runs a test to find out how many accounts meet the criteria without performing the action. The test has no effect on the accounts. This is most useful when run with the **-v** option.

-h
Displays the syntax and parameters for this utility.

Results

This utility generates notification events when a balance crosses a credit threshold. It logs errors in the log file, which is either in the directory from which the utility was started or in a directory specified in the configuration file.

pin_state_change

Use this utility to perform bulk service state transitions based on the state expiration time. See "Managing Custom Service Life Cycles".

Location

*BRM_home***/bin**

The configuration file for this utility is located in the *BRM_home*/**apps/pin_state_change** directory. Run **pin_state_change** from that directory.

Syntax

```
pin_state_change [-help] [-state] [-service] [-verbose]
```

Parameters

-help
Displays the syntax and parameters for this utility.

-state
Performs state transitions for services in this state only. Use the name of a state defined in the **<NAME>** element in a **config_lifecycle_states.xml** file.

-service
Performs state transitions for this service type only. Use the name of a BRM service type (***/service/****) that uses a custom life cycle.

-verbose
Displays information about successful or failed processing as the utility runs.

Results

The log file specified in the utility's configuration (**pin.conf**) file records the following:

- The number of services for which this utility made a state change
- The number of state changes that were successful