# Oracle® Communications Billing and Revenue Management

Managing Customers

Release 7.5

**E16700-16**

April 2017

ORACLE®

Oracle Communications Billing and Revenue Management Managing Customers, Release 7.5

E16700-16

# Contents

## 3 Customizing Registration

## 4 Setting Up Customer Self Care with Self-Care Manager

## 5 Sending Registration Information to Customers



## 6 Managing Customer Contact Information



## 7 Managing Customer Authentication

## 8 Creating and Managing Customer Segments



## 9 Managing Customer Billing Information

## 10 Managing Business Profiles



## 11 Managing Customers' Services and Products

## 12 Changing Account and Service Status

## 13 Managing Service Life Cycles



## 14 Managing Customers' Subscription-Level Services

## 15 Working with Profile Sharing Groups



## 16 Sending Email to Customers Automatically



## 17 Managing System and Account Currencies

## 20 Managing Brands



# Part III Migrating Customer Accounts

## 21 Understanding Conversion Manager

## 22 Installing and Configuring Conversion Manager



## 23 Mapping Legacy Data to the BRM Database



## 24 Migrating Data by Using New and Extended Storable Classes

## 25 Loading Legacy Data into the BRM Database



## 26 Migrating Legacy Data into BRM Table Partitions



# Part IV Customer Management Utilities

## 27 Conversion Manager Utilities

## 28 Customer Management Utilities

# Preface

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

## Audience

This document is intended for developers and system administrators.

## Accessing Oracle Communications Documentation

BRM documentation and additional Oracle documentation; such as Oracle Database documentation, is available from Oracle Help Center:

http://docs.oracle.com

Additional Oracle Communications documentation is available from the Oracle software delivery Web site:

https://edelivery.oracle.com

## Documentation Accessibility

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

### Access to Oracle Support

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

## Document Revision History

The following table lists the revision history for this book.

| Version | Date | Description |
|---|---|---|
| E16700-01 | November 2011 | Initial release. |

| Version | Date | Description |
|---|---|---|
| E16700-02 | May 2012 | Documentation updates for BRM 7.5 Patch Set 1.<br>■ Added information about the **EventAdjustmentsDuringCancellation** business parameter.<br>■ Added the following for custom service life cycles:<br>Managing Service Life Cycles chapter<br>pin_state_change section |
| E16700-03 | December 2012 | Documentation updates for BRM 7.5 Patch Set 3.<br>■ Made minor formatting and text changes.<br>■ Added the "Creating Control Files for Custom Event Tables in a Virtual Column-Enabled System" section.<br>■ Added documentation on transferring pending scheduled actions during subscription transfers (in the "Managing Customers' Subscription-Level Services" chapter). |
| E16700-04 | March 2013 | Documentation updates for BRM 7.5 Patch Set 4.<br>■ Replaced multidatabase information with multischema information. |
| E16700-05 | August 2013 | On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5.<br>Documentation added for HP-UX IA64. |
| E16700-06 | October 2013 | Documentation updates for BRM 7.5 Patch Set 6.<br>■ Added the "Transferring Subscription Services Between Accounts in Multiple Schemas" section. |
| E16700-07 | February 2014 | Documentation updates for BRM 7.5 Patch Set 7.<br>■ Made minor formatting and text changes. |
| E16700-08 | May 2014 | Documentation updates for BRM 7.5 Patch Set 8.<br>■ Made minor formatting and text changes.<br>■ Updated the "Creating Partitions for Your Legacy Data" section.<br>■ Updated the "Changing a Customer's Payment Method" section.<br>■ Added the "About Changing a Closed Child Account's Bill Unit to the Nonpaying (Subordinate) Payment Method" section.<br>■ Added the "Setting a Service-Level Balance Group" section.<br>■ Added the "Setting an Account-Level Billinfo" section.<br>■ Added the "Example of Migrating Hierarchical Bill Units within an Account" section. |

| Version | Date | Description |
|---|---|---|
| E16700-09 | August 2014 | Documentation updates for BRM 7.5 Patch Set 9.<br>■ Documentation added for RADIUS Manager.<br>■ Added information about SEPA in the following sections:<br>"Customizing Payment Method Data Preparation"<br>"Customizing Payment Method Validation"<br>"Customizing Payment Methods"<br>■ Added the "About Securing Sensitive Customer Data with Masking" section. |
| E16700-10 | October 2014 | Documentation updates for BRM 7.5 Patch Set 10.<br>■ Updated the following sections:<br>About Finding Accounts<br>Uninstalling Self-Care Manager<br>Changing Credit Card Information<br>Changing Purchase, Usage and Cycle Start Time for Reactivated Products and Discounts<br>■ Added the "Replacing Credit Card Numbers with Tokens at Registration" section. |
| E16700-11 | January 2015 | Documentation updates for BRM 7.5 Patch Set 11.<br>■ Updated the "Changing the Purchase, Usage, and Cycle Start and End Times" section. |
| E16700-12 | June 2015 | Documentation updates for BRM 7.5 Patch Set 12.<br>■ Made minor formatting and text changes. |
| E16700-13 | August 2015 | Documentation updates for BRM 7.5 Maintenance Patch Set 1.<br>■ Added the "Storing a Summary of Activities Performed on an Account" section.<br>■ Updated the following sections:<br>Specifying the ZIP Code Format<br>Adding or Changing a VAT Registration Number<br>About Brand-Aware Data |
| E16700-14 | December 2015 | Documentation updates for BRM 7.5 Patch Set 14.<br>■ Updated the following sections:<br>Creating an Account<br>Masking Sensitive Data in Logs When Using Client Applications<br>Software Requirements<br>Example of Migrating Hierarchical Accounts |
| E16700-15 | August 2016 | Documentation updates for BRM 7.5 Patch Set 16.<br>■ Added the "Audit Tables Affected by the Conversion Process" section.<br>■ Updated the following sections:<br>Deleting Accounts<br>Common pin_cmt Utility Error Messages |

| Version | Date | Description |
|---|---|---|
| E16700-16 | April 2017 | Documentation updates for BRM 7.5 Patch Set 18.<br>■ Added the "Improving Product Purchase Performance" section. |

# Part I

## Registering and Managing Customers

Part I describes the registration and management of customers. It contains the following chapters:

# 1

# About Registering Customers

This chapter provides an overview of registering a customer in Oracle Communications Billing and Revenue Management (BRM).

## About Registration

When you register a customer, you create a customer account in the BRM database. Registration typically follows this process:

1. The customer calls the customer service representative (CSR). See "Ways to Register Customers".
2. The customer chooses a plan.
3. The customer provides information such as name, address, and credit card number. In addition, a CSR might give discounts to the customer or collect billing information such as the name of a billing contact.
4. The account is created in the BRM database. At account creation, a number of events occur:
   - The customer information is checked by BRM to see whether it is valid and to make sure that all required information has been supplied. See "Specifying How to Validate Customer Contact Information".
   - If the customer entered a credit card number, it is validated by a credit card processing service. See "Credit Card Validation and Charge Options".
   - If there is a purchase fee, the customer is automatically charged.
   - A welcome message is automatically sent to the customer. See "Sending Registration Information to Customers".

## Ways to Register Customers

You can register customers by using the following methods:

- Have CSRs create accounts by using Customer Center.

  **Note:** In Customer Center, you can backdate the account creation to a date earlier than the current date.

- Give customers calling cards or voucher cards, which they use to access accounts you have already created in your BRM database.

### Choosing a Registration Method

When setting up registration, consider the following:

- You can implement both types of registration: by CSRs, and by the Web.
- Web-based registration takes longer to set up, but it is more cost effective.
- Complex plans are best offered by a CSR, who can explain the plans.
- If you use account groups, you might need a CSR to assign accounts to groups.

## Ways to Implement a Web Interface

You can implement account access in the following ways:

- Use Self-Care Manager. You can customize the appearance of the default Web pages by modifying Java Server Pages (JSPs). A programmer can add functionality by modifying Self-Care Manager's source code.

  See "Setting Up Customer Self Care with Self-Care Manager".

- Create a Web interface by using the Portal Communications Module (PCM) library. If you use the PCM library, your Web server must run on a platform that supports BRM.

## About CSR Registration

When a customer registers by phone, your CSR creates the account by using Customer Center.

See "Customizing Registration".

The CSR can perform additional account management tasks after the account is created; for example:

- Add the new customer to an account group.
- Give the customer additional discounts on products. See "Managing Customers' Services and Products".
- Change the account status or service status.

## Sending Messages to Your Customers at Registration

Through a CSR, you can configure your system to automatically send your customers messages during and after registration. These messages might include:

- Information about special offers.
- The customer's connection settings, such as news server address and mail server address.
- A list of phone numbers for dialup connections.

See "Sending Registration Information to Customers" for more information on sending messages to your customer.

## About Business and Consumer Accounts

Customer Center includes two different account creation methods:

- Use the business account method if you are creating an account for someone who runs a business, such as an Internet service provider (ISP) or an E-Commerce business. The business account method includes billing setup information such as billing cycle and accounting type.
- Use the consumer account creation method to create accounts for consumers. The consumer account creation method has fewer fields, which streamlines the account creation process.

## Modifying Products at Account Creation

CSRs can modify products when registering customers in Customer Center. See "Modifying Products" for more information on what you can customize.

**Note:** You specify if a deal is customizable at account creation when you create the price list in Pricing Center.

## Setting the Status of a New Product

When adding a deal or creating an account during registration, you can specify the status of each product. For example, if using the product requires hardware setup, you can inactivate the product until the setup is complete. After the hardware setup is complete, you can activate the product by using Customer Center. The customer is not billed for the product while it is inactivated.

**Note:** You cannot inactivate a product after it has been purchased. However, you can inactivate a service.

## How BRM Creates Accounts

The recommended opcode for creating accounts is the PCM_OP_CUST_COMMIT_CUSTOMER opcode. This is a wrapper opcode that performs all the tasks necessary to create an active and billable account in the database.

When you use PCM_OP_CUST_COMMIT_CUSTOMER, the account creation occurs within a transaction. This ensures that the account is created before sending the customer a welcome email. Using PCM_OP_CUST_COMMIT_CUSTOMER also creates a notification event that alerts Pipeline Manager to a new account.

**Note:** When PCM_OP_CUST_COMMIT_CUSTOMER is called by JCA Resource Adapter in XA Transaction mode, account creation occurs within a transaction started by a global transaction manager. For details, see "PCM_OP_CUST_COMMIT_CUSTOMER" in *BRM Developer's Reference*.

## Creating an Account

To create an account, use PCM_OP_CUST_COMMIT_CUSTOMER.

The following procedure describes the account creation process:

1. PCM_OP_CUST_COMMIT_CUSTOMER calls the PCM_OP_CUST_CREATE_CUSTOMER opcode.

   **Note:** To backdate an account's creation time, pass the backdate time in the PIN_FLD_END_T field of PCM_OP_CUST_COMMIT_CUSTOMER. The value of PIN_FLD_END_T is copied to the PIN_FLD_EFFECTIVE_T field internally and is then passed to the appropriate opcodes to complete the account creation process.

2. PCM_OP_CUST_CREATE_CUSTOMER calls the PCM_OP_CUST_PREP_CUSTOMER opcode to validate the registration information.
3. PCM_OP_CUST_PREP_CUSTOMER follows all the steps for creating an account object but does not commit the object to the database:
   - Opens a transaction.
   - Creates an **/account** storable object.
   - Sets account credit limits.
   - Purchases a deal.
   - Creates a **/service** storable object.

     If the deal purchased for the account is a required deal, the **/service** object's PIN_FLD_TYPE value is set to PIN_BILL_SERVICE_REQUIRED. This establishes the service as a required service in the account.

   If PCM_OP_CUST_PREP_CUSTOMER is unsuccessful in any of these operations, it exits and calls a base opcode to cancel the transaction. This ensures that no changes are made to the database.

   If PCM_OP_CUST_PREP_CUSTOMER successfully validates the registration data, it calls a base opcode to cancel the transaction, ensuring that the customer objects are not created, and then returns the validated registration information in the output flist to PCM_OP_CUST_CREATE_CUSTOMER.

   If PCM_OP_CUST_PREP_CUSTOMER cannot open a local transaction, or a transaction is already open, the transaction is stopped and PCM_OP_CUST_PREP_CUSTOMER exits without validating the registration information.

   In addition, PCM_OP_CUST_PREP_CUSTOMER calls the PCM_OP_PYMT_POL_SPEC_VALIDATE policy opcode to determine whether a customer's payment information needs to be validated during registration. If validation is required, PCM_OP_CUST_PREP_CUSTOMER calls the PCM_OP_PYMT_VALIDATE opcode to perform the operation. If validation fails, the results are transformed to be consistent with the results used to describe registration failure results and are then returned on the output flist. See "Changing How BRM Handles Paymentech Address Validation Return Codes" in *BRM Configuring and Collecting Payments*.
4. If the data is valid, PCM_OP_CUST_CREATE_CUSTOMER calls the PCM_OP_CUST_CREATE_ACCT opcode to create the **/account** object.

   PCM_OP_CUST_CREATE_ACCT creates a generic **/account** object. The account is constructed in the following actions, all of which are performed inside the

transaction started by PCM_OP_CUST_COMMIT_CUSTOMER or the global transaction manager:

- Calls the PCM_OP_CUST_CREATE_BILLINFO opcode to create one or more **/billinfo** storable objects and propagates the bill unit with billing information. See "Creating /billinfo Objects" in *BRM Configuring and Running Billing*.
- Calls the PCM_OP_CUST_CREATE_BAL_GRP opcode to create one or more **/balance_group** storable objects and link the balance groups to the account and the appropriate **/billinfo** object. See "Creating Balance Groups" in *BRM Managing Accounts Receivable*.
- Calls the PCM_OP_CUST_CREATE_PAYINFO opcode to create a **/payinfo** object and link the payment information to the appropriate **/billinfo** object. See "Customizing Customer Payment Information".
- Calls the PCM_OP_CUST_CREATE_PROFILE opcode to create a **/profile** object if the input flist contains profile information. See "Managing and Customizing Profiles".
- Calls the PCM_OP_CUST_SET_NAMEINFO opcode to set the contact information. See "Managing Customer Contact Information".
- Calls the PCM_OP_CUST_SETUP_TOPUP opcode to set up top-up information, if any, for the account. See "How BRM Sets Up Top-Up Information for an Account" in *BRM Configuring and Collecting Payments*.
- Calls the PCM_OP_CUST_SET_STATUS opcode to set the account status to active. See "Setting Account, Service, and Bill Unit Status by Using Your Custom Application".

5. PCM_OP_CUST_CREATE_CUSTOMER calls the PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY opcode to validate that the deals passed from the input flist do not violate any deal dependencies.
6. PCM_OP_CUST_CREATE_CUSTOMER calls the PCM_OP_CUST_CREATE_SERVICE opcode to create **/service** objects. See "Creating Services".

7. PCM_OP_CUST_CREATE_CUSTOMER calls the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode to purchase deals. See "Purchasing Deals."

   When calling PCM_OP_SUBSCRIPTION_PURCHASE_DEAL, PCM_OP_CUST_CREATE_CUSTOMER populates the PIN_FLD_DEAL_INFO field in the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL input flist only when the following is true for the input flist of PCM_OP_CUST_CREATE_CUSTOMER:

   - At the PIN_FLD_ACCTINFO level:
     - PIN_FLD_DEAL_OBJ is not populated.
     - PIN_FLD_DEAL_INFO is populated, including the PIN_FLD_PRODUCTS array and its PIN_FLD_STATUS field.
   - At the PIN_FLD_SERVICES level:
     - PIN_FLD_DEAL_OBJ is not populated.
     - PIN_FLD_DEAL_INFO field is populated, including the PIN_FLD_PRODUCTS array and its PIN_FLD_STATUS field.
   - At the PIN_FLD_SERVICES > PIN_FLD_DEALS level:
     - PIN_FLD_DEAL_OBJ is not populated.
     - PIN_FLD_DEAL_INFO field is populated, including the PIN_FLD_PRODUCTS array and its PIN_FLD_STATUS field.

8. PCM_OP_CUST_COMMIT_CUSTOMER creates the customer's collections profile at the time of customer account creation. See "Creating Customer's Collections Profiles at the Time of Customer Account Creation".

9. Before committing the transaction, PCM_OP_CUST_COMMIT_CUSTOMER calls the PCM_OP_CUST_POL_PRE_COMMIT policy opcode. See "Adding Custom Account Creation Steps before the Account Is Committed".

10. After committing the transaction, PCM_OP_CUST_COMMIT_CUSTOMER calls the PCM_OP_CUST_POL_POST_COMMIT policy opcode. See "Adding Custom Account Creation Steps after the Account Is Committed".

11. PCM_OP_CUST_COMMIT_CUSTOMER:

    - Passes credit card information to PCM_OP_PYMT_VALIDATE and PCM_OP_PYMT_COLLECT, which collect any credit card payments charged at account creation and validate the credit card information returned by the payment processor.
    - If you use multiple schemas, creates the **/uniqueness** object.
    - Calls the PCM_OP_CUST_POL_GET_CONFIG policy opcode to get information used by client applications.
    - Calls the PCM_OP_CUST_SET_BAL_GRP opcode to create balance groups. See "Creating Balance Groups" in *BRM Managing Accounts Receivable*.

    **Note:** By default, BRM does not log events that do not have a balance impact. You can specify to log all account-creation events. See "Logging Non-Currency Events" in *BRM System Administrator's Guide*.

### Creating Customer's Collections Profiles at the Time of Customer Account Creation

Use the PCM_OP_CUST_COMMIT_CUSTOMER opcode to create a customer's collections profile at the time of customer account creation.

Customer Center calls this opcode when creating a customer account.

As input, this opcode takes **/profile/collections_params** as the Portal object ID (POID) type in the PIN_FLD_PROFILE_OBJ field and the collections parameter details in the PIN_FLD_COLLECTIONS_PARAMS array to create the collections profile. This opcode uses the PIN_FLD_BILLINFO_ID field to identify the **/billinfo** object in case of multiple bill units.

Sample input flist:

```
# PCM_OP_CUST_COMMIT_CUSTOMER input flist
0 PIN_FLD_POID                  POID [0] 0.0.0.1 /plan 102289 0
0 PIN_FLD_LOCALES               ARRAY [1] allocated 20, used 1
1  PIN_FLD_LOCALE                STR [0] "en_US"
.
.
0 PIN_FLD_PAYINFO               ARRAY [0] allocated 20, used 5
1  PIN_FLD_NAME                  STR [0] "Invoice1"
0 PIN_FLD_PROFILES              ARRAY [0]
1  PIN_FLD_PROFILE_OBJ           POID [0] 0.0.0.1 /profile/collections_params -1 0
1  PIN_FLD_INHERITED_INFO        SUBSTRUCT [0]
2   PIN_FLD_COLLECTIONS_PARAMS    ARRAY [0] allocated 4, used 4
3    PIN_FLD_BILLINFO_ID            STR [0]"Bill Unit(1)" #(other collections
attributes fields, say VF_FLD_DD_REJECT_CODE)
3    PIN_FLD_BILLINFO_OBJ           POID [0] NULL
2   PIN_FLD_COLLECTIONS_PARAMS    ARRAY [1] allocated 4, used 4
3    PIN_FLD_BILLINFO_ID            STR [0]"Bill Unit(2)"
3    PIN_FLD_BILLINFO_OBJ           POID[0] NULL
```

## About Creating Multiple Accounts in One Transaction

You can create multiple BRM accounts in one transaction with JCA Resource Adapter. This is useful, for example, when a single order in Oracle Communications Order and Service Management (OSM) produces multiple new customers. It enables Oracle Application Integration Architecture (Oracle AIA) to maintain the cross-references between the OSM order and its associated BRM accounts.

For more information, see the description on JCA Resource Adapter transaction management in *BRM JCA Resource Adapter*.

## About Creating Accounts with Backdated Services or Resources

By default, BRM does not allow accounts to be created with the services or resources creation date backdated beyond the account or service creation date.

You can create accounts with the services or resources creation date backdated beyond the account or service creation date by enabling a business parameter in the **subscription** instance of the **/config/business_params** object.

### Enabling the Business Parameter to Allow Backdated Services and Resources

Use the **pin_bus_params** utility to enable the **SubsDis74BackDateValidations** business parameter in the **subscription** instance of the **/config/business_params** object.

To create accounts with backdated services or resources:

1. Go to the *BRM_Home***/sys/data/config** directory, where *BRM_Home* is the directory in which you installed the BRM software.

2. Use the following command to create an editable XML file of the **subscription** instance from 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. To place this file in a different directory, specify the full path as part of the file name.

3. Search the XML file for the following line:

   ```
   <SubsDis74BackDateValidations>disabled</SubsDis74BackDateValidations>
   ```

4. Change **disabled** to **enabled**.

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

5. Save this updated file as **bus_params_subscription.xml**.

6. Use the following command to load this change into the **/config/business_params** object in the BRM database.

   ```
   pin_bus_params -r BusParamSubscription 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 the description for **pin_bus_params** in *BRM System Administrator's Guide*.

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

   For more information on reading objects by using Object Browser, see *BRM Managing Customers*. For instructions on using the **testnap** utility, see *BRM Developer's Guide*.

8. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

9. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in the *BRM System Administrator's Guide*.

## About Backdated Account Creation

When you backdate the account creation, the account becomes effective as of the new backdated date. BRM does the following:

- Sets the PIN_FLD_EFFECTIVE_T field in the **/account** object to the backdated date.
- Sets the purchase, usage, or cycle start date to the backdated date unless these dates are explicitly set to a different date.
- Sets the billing day of month (DOM) to the backdated date unless the **actg_dom** entry in the CM configuration file (**pin.conf**) is used.

BRM does not allow backdating the account creation to a date prior to the general ledger (G/L) posting date. See "About Backdating Beyond the G/L Posting Date".

### How Cycle Fees Are Calculated for Backdated Account Creation

BRM uses the PCM_OP_SUBSCRIPTION_CYCLE_FORWARD and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS opcodes to calculate cycle forward and cycle arrears fees for the subscription operations.

When you backdate the creation of an account that has cycle forward events, the cycle forward fee is applied for the first cycle just as it would be if the account were to be created on the backdated date. The remaining cycles, until the current cycle, are charged only during the next bill run.

For example, consider that a customer places a request for an account creation on September 1, but the entry is recorded in the BRM system on November 1. So, on November 1, the entry is backdated to September 1. The account owns a product with cycle forward charges of $9.95 and the proration set to **Charge based on amount used**.

BRM calculates the cycle forward fees as follows:

- Immediately charges the cycle forward fee for the period of September 1 to October 1.
- The bill for October 1 contains a monthly cycle fee charge of $19.90 ($9.95 for the period of September 1 to October 1 and $9.95 for the period of October 1 to November 1).
- The bill for November 1 contains a monthly cycle fee charge of $9.95 for the period of November 1 to December 1.

When you backdate the creation of an account that has cycle arrears events, cycle arrears fees are not generated at the time of account creation. The backdated cycles are charged during the skipped billing that gets triggered for these cycles as a result of the next bill run.

For example, consider that a customer places a request for an account creation on September 1, but the entry is recorded in BRM system on November 1. So, on November 1, the entry is backdated to September 1. The account owns a product with cycle arrears charges of $9.95 and the proration set to **Charge based on amount used**.

BRM calculates the cycle arrears fee as follows:

- The bill for October 1 contains a monthly cycle fee charge of $9.95 for the period of September 1 to October 1.
- The bill for November 1 contains a monthly cycle fee charge of $9.95 for the period of October 1 to November 1.

**Important:**

- BRM does not prevent backdated creation, modification, or deletion of the discount sharing group and charge sharing group. However, the resource sharing is not effective from the backdated date. For example, on October 1, the group owner of a discount sharing group is modified and backdated to January 1 to include a group member account. The group member cannot enjoy discounts owned by the group owner with effect from January 1. For discounts to be effective from January 1, you must run rerating on all the accounts in the sharing group.
- BRM does not support backdating the change of the billing DOM. For example, you change the DOM from 5 of every month to 1 of every month. You cannot backdate this change to an earlier date.

### Backdating Account Creation

To backdate the account creation, pass the desired date in the PIN_FLD_END_T field of the PCM_OP_CUST_COMMIT_CUSTOMER opcode. You can also backdate the account creation using Customer Center. See the Customer Center Help.

## Modifying an Account

To modify an account, use the PCM_OP_CUST_UPDATE_CUSTOMER and PCM_OP_CUST_MODIFY_CUSTOMER opcodes.

PCM_OP_CUST_UPDATE_CUSTOMER calls other opcodes to add, modify, or delete the following data:

- Contact information
- Payment information: see "Changing a Bill's Payment Method".
- Bill units
- Top-up information
- Balance groups
- Profiles
- Locale

PCM_OP_CUST_MODIFY_CUSTOMER adds a service by adding a deal to the account. PCM_OP_CUST_MODIFY_CUSTOMER adds a deal by calling standard opcodes to perform these operations:

- If the **validate_deal_dependencies** entry is enabled in the CM **pin.conf** file, PCM_OP_CUST_MODIFY_CUSTOMER verifies that any set deal dependencies are valid. If not, the operation is disallowed. See "Defining Deal Dependencies".
- Purchase the deal associated with the plan. See "Purchasing Deals".
- Set the account credit limit for each resource of the added plan. See "How BRM Handles Consumption Rules and Credit Limits".
- Create service objects associated with the plan.
  - If a service with an add-on plan is added, and the **ValidateDiscountDependency** flag is enabled in the **/config/business_params** object, PCM_OP_CUST_MODIFY_CUSTOMER checks for any mutually exclusive relationships between discounts associated with the add-on plan and any discounts already owned by the account. If any such relationship is found, the service addition is not continued.

    **Note:** Discount-to-discount exclusion rules are not validated. PCM_OP_CUST_MODIFY_CUSTOMER validates only plan-to-discount exclusion rules at purchase time.

  - If a subscription service is being added, PCM_OP_CUST_MODIFY_CUSTOMER verifies that it is not a member of another subscription group.
  - If the *account status* is inactive and the provisioning flag is not set, an error value is returned and PCM_OP_CUST_MODIFY_CUSTOMER exits without modifying the account.
  - If a subscription member service is being created, PCM_OP_CUST_MODIFY_CUSTOMER verifies that the subscription service is not inactive or closed. If

the member service requires its own balance group, PCM_OP_CUST_MODIFY_CUSTOMER creates the balance group. Otherwise, it associates the member service with the subscription service's balance group.

- Optionally associate **/device** objects with the services. See "Managing Devices with BRM" in *BRM Developer's Guide*.
- Create notification events to mark the beginning and end of its execution. See "About Event Notification" in *BRM Developer's Guide*.

You can use other opcodes to modify accounts, such as PCM_OP_CUST_MODIFY_PAYINFO and PCM_OP_CUST_MODIFY_PROFILE.

PCM_OP_CUST_MODIFY_CUSTOMER includes a hook to the PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER policy opcode, which you can use to export customer data to an external or legacy system for processing. See "Customizing Registration".

Table 1–1 lists the opcodes to which PCM_OP_CUST_MODIFY_CUSTOMER passes information from arrays in its input flist. The opcodes, in turn, use the information to create or modify objects:

***Table 1–1 Opcodes and Information Received to Create or Modify Objects***

| This Opcode | Uses Information from This Array | To Create or Modify This Object |
|---|---|---|
| PCM_OP_CUST_SET_NAMEINFO | PIN_FLD_NAMEINFO<br>PIN_FLD_PHONES (subarray) | **/account** |
| PCM_OP_CUST_SET_BAL_GRP | PIN_FLD_ACCTINFO<br>PIN_FLD_BAL_INFO (subarray) | **/balance_group** |
| PCM_OP_CUST_SET_BILLINFO | PIN_FLD_BILLINFO | **/billinfo** |
| PCM_OP_CUST_SET_PAYINFO | PIN_FLD_PAYINFO<br>PIN_FLD_CC_INFO (subarray)<br>PIN_FLD_DD_INFO (subarray)<br>PIN_FLD_INV_INFO (subarray) | **/payinfo/cc**<br>**/payinfo/dd**<br>**/payinfo/invoice** |
| PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER | PIN_FLD_ACCTINFO<br>PIN_FLD_BAL_INFO (subarray)<br>PIN_FLD_BILLINFO | **/balance_group**<br>**/billinfo** |
| PCM_OP_CUST_SET_LOCALE | PIN_FLD_LOCALE | **/account** |
| PCM_OP_CUST_CREATE_PROFILE<br>PCM_OP_CUST_MODIFY_PROFILE | PIN_FLD_PROFILES | **/profile** |
| PCM_OP_CUST_SET_TOPUP | PIN_FLD_TOPUP_INFO<br>PIN_FLD_GROUP_TOPUP_INFO (subarray)<br>PIN_FLD_GROUP_TOPUP_LIMITS (subarray)<br>PIN_FLD_GROUP_TOPUP_MEMBERS (subarray) | **/topup**<br>**/group/topup** |

If the information is successfully updated, the opcode returns the POID of the **/event** storable objects created in the PIN_FLD_RESULTS array of the output flist. Otherwise, it returns the PIN_FLD_FIELDS array specifying the failing field.

If the input flist array value for PIN_FLD_NAMEINFO is set to NULL, the corresponding element ID is deleted from the NAMEINFO table in the **/account** object.

### Changing a Bill's Payment Method

Use PCM_OP_CUST_UPDATE_CUSTOMER to change the payment method for an account's bill unit in the following ways:

- To change an existing payment method for a bill unit, pass the POID of the **/payinfo** object in the PIN_FLD_ PAYINFO_OBJ field in the PIN_FLD_BILLINFO array.

  For example:

```
0 PIN_FLD_POID POID                [0] 0.0.0.1 /account 23304551863 227
0 PIN_FLD_ACCOUNT_OBJ POID         [0] 0.0.0.1 /account 23304551863 227
0 PIN_FLD_PROGRAM_NAME STR        [0] "program_name"
0 PIN_FLD_BILLINFO ARRAY          [0] allocated 20, used 3
1    PIN_FLD_BILLINFO_ID STR      [0] "Account Bill"
1    PIN_FLD_PAY_TYPE ENUM        [0] 10005
1    PIN_FLD_BILLING_SEGMENT ENUM [0] 0
1    PIN_FLD_PAYINFO_OBJ POID      [0] 0.0.0.1 /payinfo/cc 4780 0
...
```

- To add a new payment method for an existing bill unit, you can create the **/payinfo** object and associate it with the **/billinfo** object in the same call by doing the following:
  - Include a PIN_FLD_PAYINFO_ARRAY field (outside of the PIN_FLD_ BILLINFO array) and add an array element for the new **/payinfo** object.
  - In the PIN_FLD_BILLINFO array, include the PIN_FLD_PAY_TYPE field with the new payment method. Specify a NULL value for the PIN_FLD_PAYINFO array and make sure the array element ID matches the element ID of the new **/payinfo** object in the PIN_FLD_PAYINFO array that is outside the billinfo array.

  For example:

```
0 PIN_FLD_POID POID                [0] 0.0.0.1 /account 23304551863 227
0 PIN_FLD_ACCOUNT_OBJ POID         [0] 0.0.0.1 /account 23304551863 227
0 PIN_FLD_PROGRAM_NAME STR        [0] "program_name"
0 PIN_FLD_BILLINFO ARRAY          [0] allocated 20, used 3
1    PIN_FLD_BILLINFO_ID STR      [0] "Account Bill"
1    PIN_FLD_PAY_TYPE ENUM        [0] 10005
1    PIN_FLD_BILLING_SEGMENT ENUM [0] 0
1    PIN_FLD_PAYINFO ARRAY         [1] NULL
0 PIN_FLD_PAYINFO ARRAY            [1] allocated 20, used 20
1    PIN_FLD_POID POID            [0] 0.0.0.1 /payinfo/cc -1 0
...
```

  In the above example, the new **/payinfo** object is created and associated with the **/billinfo** object specified in the input flist.

**Note:** You must specify either the PIN_FLD_PAYINFO array or the PIN_FLD_ PAYINFO_OBJ field in the PIN_FLD_BILLINFO array. However, because these fields are mutually exclusive and you cannot specify both, they are classified as optional.

## Creating Services

To create a service object, use PCM_OP_CUST_CREATE_SERVICE.

If the PIN_FLD_SUBSCRIPTION_OBJ field in the input flist specifies the POID of a subscription's service object, the service being added to the customer's account will be a member of the subscription service's group.

PCM_OP_CUST_CREATE_SERVICE creates the service by doing the following inside a transaction:

- Calls the PCM_OP_CUST_INIT_SERVICE opcode to create and initialize the **/service** object.
- Calls the PCM_OP_CUST_SET_PASSWD opcode to encrypt and set a password (if one is supplied) in the **/service** object. If a password is not provided, one is generated. See "Customizing Passwords".
- Stores any client-supplied inherited information in the **/service** object. See "Creating Customization Interfaces" in *BRM Developer's Guide*.
- Calls the PCM_OP_CUST_SET_STATUS opcode to set the service status. See "Setting Account, Service, and Bill Unit Status by Using Your Custom Application".
- Calls the PCM_OP_DEVICE_ASSOCIATE opcode to associate **/device** objects with the service. See "Managing Devices with BRM" in *BRM Developer's Guide*.

## Modifying Services

To modify a service, use the PCM_OP_CUST_UPDATE_SERVICES opcode. For most services, this wrapper opcode calls the PCM_OP_CUST_MODIFY_SERVICE opcode to set, change, or delete extended service information.

PCM_OP_CUST_UPDATE_SERVICES takes as input the POID of the **/account** object, the name of the calling program, and an array containing a list of services to be updated. Additional inputs depend on the type of services being updated. Services are processed in the order dictated by the element value passed in the PIN_FLD_ SERVICES array.

PCM_OP_CUST_UPDATE_SERVICES performs these operations:

1. Creates an **/event/notification/service/pre_change** event.
2. If login or alias list information is specified, calls the PCM_OP_CUST_SET_LOGIN opcode to update the service login or alias list with the value specified.
3. If password information is specified, calls the PCM_OP_CUST_SET_PASSWD opcode to update the service password with the value specified.
4. If service status information is specified, calls the PCM_OP_CUST_SET_STATUS opcode to update the status of the service with the value specified.
5. If inherited fields are specified, calls PCM_OP_CUST_MODIFY_SERVICE for most services to update the values for the inherited fields.

6. If one or more PIN_FLD_DEVICES arrays are specified, calls the PCM_OP_DEVICE_ASSOCIATE opcode to disassociate or associate the device. Disassociations are processed before associations.
7. Creates an **/event/notification/service/post_change** event.

If successful, the PCM_OP_CUST_UPDATE_SERVICES output flist contains the following:

- The POID of the **/account** object passed in.
- A services array containing the following:
  - The POIDs of the **/service** objects modified.
  - A results array containing the POID of the **/event** objects created to record the changes. The index values returned in the PIN_FLD_RESULTS array correspond to the input flist values as shown in Table 1–2:

***Table 1–2 Indexes and Input Flist Values in PIN_FLD_RESULTS***

| Index Value | Input Flist Value |
|---|---|
| 0 | Login |
| 1 | Password |
| 2 | Status |
| 3 | Inherited information |

If an error is encountered in the service data, PCM_OP_CUST_UPDATE_SERVICES:

- Continues to process all of the data, so that all erroneous data is uncovered.
- Rolls back the transaction. Any updates already made to the information are undone.
- Returns an error in the error buffer.

After the object is modified, PCM_OP_CUST_MODIFY_SERVICE creates the **/event/notification/service** event to notify listeners about the change to the **/service** object. The output flist contains the POID of the modified **/service** object. In case of an error, the error buffer is populated and the output flist can be ignored.

The input flist for PCM_OP_CUST_MODIFY_SERVICE contains the POID of the service object to be modified and a substruct of the inherited information to be stored. If the input flist for an inherited field of the service object contains a null entry, the array table entry is deleted from the object in the database.

## Deleting Accounts

**Caution:** *Do not* delete accounts in a production system due to the following reasons:

- The opcode does not remove audit data.
- The processing speed is slow, because the opcode must remove a lot of data associated with each account.
- If your system supports subscription service transfers, the opcode does not delete accounts in the proper order. With subscription service transfers, the source account must be deleted prior to the destination account due to **/balance_group** locking. For more information, see "About Transferring a Subscription Service to Another Subscriber".

To delete accounts in a test system, use the PCM_OP_CUST_DELETE_ACCT opcode.

# 2

# About Managing Customers

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) customer management.

> **Caution:** Do not use or modify this product except as explicitly instructed in this documentation. Assumptions should not be made about functionality that is not documented or use of functionality in a manner that is not documented. Use or modification of this software product in any manner or for any purpose other than as expressly set forth in this documentation may result in voidance or forfeiture of your warranties and support services rights. Please consult your software license agreement for more details. If you have any questions regarding an intended use or modification of this product, please contact Oracle.

For more information on customer management procedures, see "Typical Customer Management Tasks".

## About Accounts

Each of your customers has an *account* in the BRM database. BRM accounts include or are linked to information about the customer's name and address, product and services, and current balance.

In addition to customer accounts, the database includes the following special-purpose accounts:

- The *root account* is used by system administrators to set permissions for customer service representatives (CSRs).

  > **Note:** BRM excludes the root account from billing. Therefore, BRM does not permit the root account to purchase services, deals, or plans for itself.

- *CSR accounts* enable CSRs to log in to the BRM system. You also use CSR accounts to track CSR activity. See "Setting Up Permissions in BRM Applications" in *BRM System Administrator's Guide*.
- *Remittance accounts* are used to track commissions for business-to-business relationships. See "Remitting Funds to Third Parties" in *BRM Configuring and Running Billing*.

- The *verification account* is used to track customer login failures. See "Tracking Customer Authentication and Authorization Records".

For more information on creating accounts, see "About Registering Customers".

## About Accounts and Services

When a customer purchases a service, BRM keeps a record of the service information for the customer's account in the BRM database. For example, the email service includes information about the customer's login name and password, the maximum message size, and the maximum number of messages allowed in a mailbox.

An account can have multiple services, such as Internet access and email. An account can also have multiple services of the same type, such as multiple email accounts.

You can create customer accounts without charging for any services. For example, you can have customers pay a monthly fee for an account with no services. In that case, a customer can add and cancel services and still be charged a consistent amount just for owning an account.

## About Account Balances

Account balances keep track of how much a customer owes, or is credited with, for each resource. For example, if a customer uses products that charge US dollars for Internet access hours, the balances track US dollars and, depending on your price list, might track Internet access hours.

Accounts can have multiple balances, which are stored in a balance group. A balance group contains a collection of sub-balances that are grouped by the type of resource they track. A resource includes more than one sub-balance when portions of the resource are valid at different times. For example, a resource of free minutes might include 300 minutes that are valid only for the current month and 1000 free minutes that never expire.

A balance group can contain any number of currency and non-currency sub-balances. The sub-balance for a currency resource is normally zero or a *debit balance*, which means the customer owes you. The sub-balance for a non-currency resource is normally zero or a *credit balance*, which means you owe the customer. A debit sub-balance is represented as a positive number. A credit sub-balance is represented as a negative number.

For example, if a customer pays $10 per month for Internet access and receives 15 free hours per month, the balances at the start of the month are:

- A currency sub-balance of 10 US dollars
- A non-currency sub-balance of -15 Internet hours

An account can also have multiple balance groups. Each balance group keeps track of how much a customer owes or is owed for one or more specific services. This means an account can own two services of the same type and each service can have its own credit limit.

For more information on how resources are tracked and updated in account sub-balances, see "About Tracking Resources in Account Sub-Balances" in *BRM Setting Up Pricing and Rating*.

# About Account Locking

Some transactional processes require that an account be locked to maintain data integrity. For these operations, the account is locked and unlocked automatically. When an account is locked, no other opcode can perform operations on the account.

# Making Notes about Customers

You can add notes to accounts to record information obtained from customers and to explain why you performed various actions. To do so, see "Creating and Reviewing Notes in Customer Center".

## Creating and Reviewing Notes in Customer Center

Customer Center categorizes 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 the following: 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, and Account/Service Status Change.

*General notes*, such as notes about the best time of day to call a customer, apply to the entire account. See "Creating General Account Notes".

Notes in all other categories apply to specific actions, such as adjustments, charges, credits, disputes, credit limit changes, write-offs, and status changes. See "Creating Notes about Specific Actions".

To review existing notes, see "Displaying Notes in Customer Center".

---

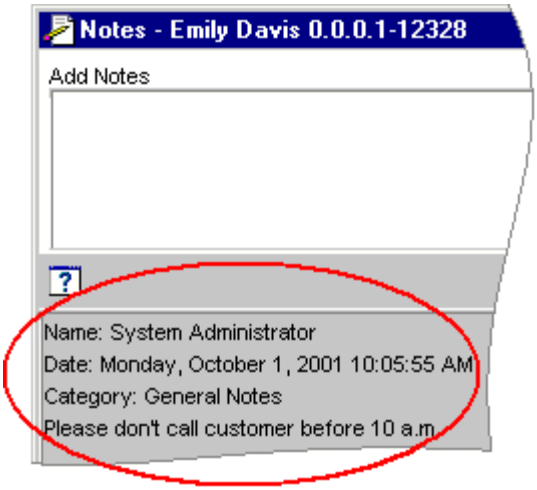
**Note:** Text entered in the **Comments** box of the Account Creation or Purchase wizard is not saved as a note. Instead, it appears in the **Comments** box in an account's **Product Detail** panel.

---

### Creating General Account Notes

Notes that apply to the entire account, such as notes about the best time of day to call a customer, are categorized as general notes, as shown in Figure 2–1:

***Figure 2–1 Example of a General Note***

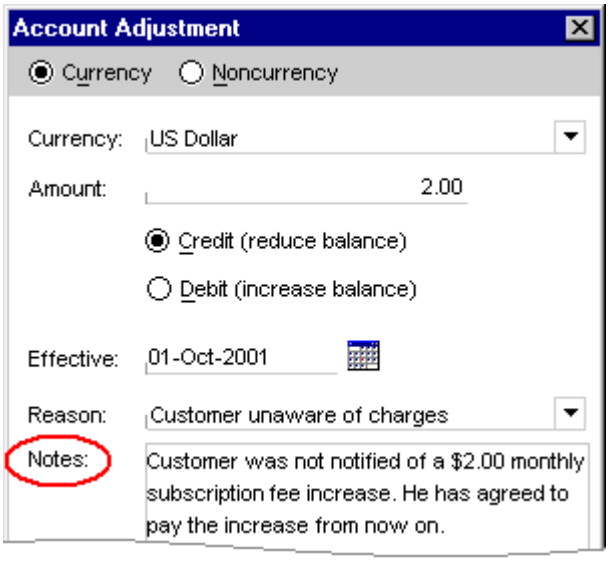


To review general notes, see "Displaying Notes in Customer Center".

### Creating Notes about Specific Actions

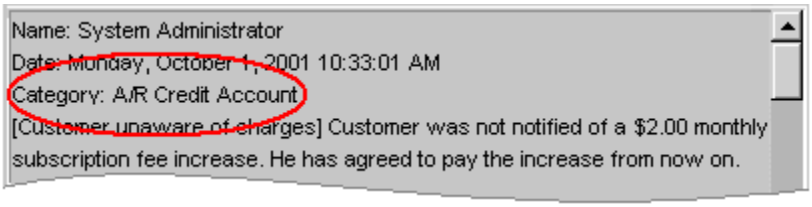
Many Customer Center dialog boxes and panels contain a **Notes** box as shown in Figure 2–2:

***Figure 2–2 Note Associated with a Specific Action***




When you enter text in a **Notes** box and save your changes, Customer Center creates a note and saves it in an action-specific category. The category is identified when the note is displayed in the Notes dialog box as shown in Figure 2–3:

***Figure 2–3 Action-specific Note Stored by Customer Center***




The category in which a note is saved depends on the action performed. For example, if the action in Figure 2–3 was a debit instead of a credit, it would be categorized as *A/R Debit Account*.

To review notes about specific actions, see "Displaying Notes in Customer Center".

> **Note:** Text entered in the **Comments** box of the Account Creation wizard or the Purchase wizard is not saved as a note. Instead, it appears in the **Comments** box in an account's **Product Detail** panel.

### Displaying Notes in Customer Center

Notes can be created in several places in Customer Center. For example, you enter general notes about an account in the Notes dialog box, and you enter notes about credits and debits in the Account Adjustment dialog box.

No matter where notes are created in Customer Center, however, they are all displayed in the same place: the Notes dialog box.

> **Note:** Text entered in the **Comments** box of the Account Creation wizard or the Purchase wizard is not saved as a note. Instead, it appears in the **Comments** box in an account's **Product Detail** panel.

To display all notes created for an account in Customer Center, see the discussion on displaying notes in the Customer Center Help.

# Storing a Summary of Activities Performed on an Account

This section describes how to enable storing a summary of activities performed on an account as News Feed in BRM.

## About News Feed

News Feed is a summary of activities performed on an account, which are stored in the BRM database. To store a summary of activities performed on a customer account, enable News Feed for each event you want a summary of. See "Enabling News Feed".

You can enable News Feed for the following events:

- **Accounts receivable (A/R)**, which includes currency adjustments, noncurrency adjustments, open disputes, closed disputes, refunds, write-offs, 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 plan or deal, bill issued, and bill issued midcycle.

## Enabling News Feed

To enable News Feed 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 News Feed:

   ```
   180  0   event_name
   ```

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

   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). See the discussion about starting and stopping the BRM system in *BRM System Administrator's Guide*.

## Loading Localized and Customized Strings for News Feed

The strings for News Feed 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 News Feed (for example, "Payment reversed"). To do so, you edit a copy of the *BRM_Home*/**sys/msgs/newsfeed/newsfeed.en_US** sample file. 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*.

### Loading Customized Strings for News Feed

To load customized strings for News Feed:

1. Open the *BRM_Home*/**sys/msgs/newsfeed/newsfeed.***locale* file in a text editor.
2. To add new strings for News Feed, 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 "About Customizing News Feed".
3. To modify existing strings for News Feed, search for the string and change the string value.
4. Save and close the file.
5. Load the strings by using the **load_localized_strings** utility.

## About Customizing News Feed

The PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode is used to process the events and updates the database with the corresponding News Feed entries. The PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode is called by the PCM_OP_PROCESS_EVENTS standard opcode.

Use the PCM_OP_ACT_POL_PROCESS_EVENTS policy opcode to customize values in News Feed and to add new events for News Feed.

As input, this opcode takes the following information:

- PIN_FLD_EVENT: Specifies the events.
- PIN_FLD_TYPE: Specifies the type of news feed, which differentiates the News Feed category from other News Feeds.
- PIN_FLD_REASON_ID: Specifies the reason code for News Feed, which links to the localized name of the News Feed category.
- PIN_FLD_MESSAGE: Specifies the summary information for News Feed. The message has the following format:

  *numeric_ID1*`;;`*numeric_ID2*`|~|`*placeholder_value1*`|~|` *placeholder_value2*

  where:

  - *numeric_IDN* is a unique identifier that maps to the reason code in the localization message. You can add multiple numeric IDs separated by **;;**.
  - *placeholder_valueN* is the value that replaces the placeholder character in the localization message for the corresponding numeric ID. The placeholder values are separated by `|~|`.

  For example:

  ```
  132;;133|~|P1-1|~|10003
  ```

  which displays the following message:

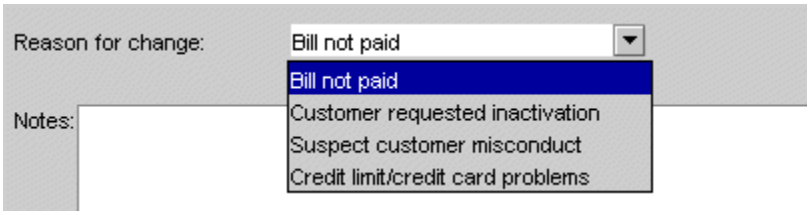
  ```
  Payment reversed
  P1-1 original payment, 10003
  ```

- PIN_FLD_FLAGS: Specifies whether to create a **/newsfeed** object:
  - When set to **1**, the opcode creates a **/newsfeed** object.
  - When set to **0**, the opcode does not create a **/newsfeed** object.

# Customizing the List of Reasons for Account Changes

When making changes to accounts, CSRs can choose from a list of reasons for making the change. Figure 2–4 shows how Customer Center displays a list of reasons for inactivating an account:

***Figure 2–4 Default Reasons for Account Changes***




Reasons for changing an account are stored in the BRM database. You can edit the list of reasons for the following changes:

- Activating or inactivating an account
- Closing an account
- Crediting, debiting, or adjusting an account
- Charging an account
- Changing credit limits

Customer Center displays the reasons in lists. You can add, edit, or delete the text for these reasons by modifying the reason objects stored in the BRM database.

To modify the reasons file, edit the *BRM_Home***/sys/msgs/reasoncodes/reasons.en_US** sample file, where *BRM_Home* is the directory in which BRM is installed. You then use the **load_localized_strings** utility to load the contents of the file into the **/strings** objects. See "load_localized_strings" in *BRM Developer's Guide*.

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

```
load_localized_strings reasons.locale
```

where *locale* is the file suffix (for example, **en_US**). If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.

**Caution:** The **load_localized_strings** utility overwrites existing **/config/map_glid** and **/config/reason_code_scope** objects. If you are updating these objects, you cannot load new general ledger (G/L) ID maps and reason code scopes only. You must load complete sets of data each time you run the utility. This is also true when using the **/strings** object, but only if you specify the **-f** parameter. Otherwise, the utility appends the new data to the object.

**Note:** If you add your own reason codes to the **reasons.***locale* file, use IDs above 100,000.

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

For more information on creating strings for this file, see "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide*.

# About Finding Accounts

In Customer Center, the first step in any customer management task is to find the account. To find an account, you search for customers based on one or more account attributes (for example, name, account number, or credit card number, or token if credit card tokenization is enabled. See the description for creating and finding accounts in the Customer Center Help.

## How Searching Handles Text Characters

For Latin 1 languages, BRM searches by using a simplified form of any entries in the **Name** and **Company Name** boxes. This means that the search ignores capital letters, accents, and diacritical marks in the search box and the account. Latin 1 languages include those of Western Europe, Canada, and the United States.

## Customizing Search

You can customize Customer Center search functions.

## Finding Customer Accounts Using Opcodes

In addition to using the base search opcodes to find accounts, you can use the PCM_OP_CUST_FIND opcode.

This opcode replaces the PCM_OP_SEARCH and PCM_OP_STEP_SEARCH opcodes. Given an account number, PCM_OP_CUST_FIND finds the **/account** storable object in question and returns the Portal object ID (POID) of the storable object and any other information that you request. If no results array is specified, the entire contents of the storable object are returned.

You can specify an array of account numbers in an input flist and have an array of results returned.

For more information on these opcodes and the **/account** storable object, see *BRM Developer's Guide*.

# About Exporting Customer Information to a Non-BRM Database

BRM can automatically export registration information to a database other than the BRM database. You might want to do this to support a legacy system or to collect additional information about customers. (A better way to collect information about customers is by using profiles.)

For more information on exporting registration information, see "Exporting Registration Information to a Non-BRM Database".

# Typical Customer Management Tasks

Most customer management tasks are accomplished by using Customer Center.

You can also set up Web-based customer self-administration so that your CSRs can focus on customer problems. See "Ways to Implement a Web Interface".

### Typical Customer Problems:

- Cannot log in to a service. Common reasons customers cannot access their accounts are:
  - They typed the wrong login name or password.

  - They forgot that passwords are case sensitive.
  - The account is inactive or closed.
  - There are credit problems.

  Possible ways to address the problem are:

  - Check the login name. See "Managing Customer Authentication".
  - Check the status of the account. See "Changing Account and Service Status".
  - Check the credit card number and expiration date. See "Changing Credit Card Information".
  - Check the credit limit. See "Changing a Customer's Credit Limit".
  - Check the amount due.
- Wrong amount on a bill. Common complaints are:
  - I was not told I would be charged for this.
  - My bill is being sent to the wrong address.
  - I did not order this.

  Possible ways to address such complaints are:

  - Cancel the service. See "Managing Services".
  - Change the billing address. See "Managing Customer Contact Information".
  - Make an adjustment. See "About Adjustments" in *BRM Managing Accounts Receivable*.
  - Open a dispute. See "About Opening and Resolving Disputes" in *BRM Managing Accounts Receivable*.
- Forgotten password or login name.

**Typical Customer Requests:**

- Change an address or phone number.
- Change an email address, password, or login name.
- Add or cancel services.
- Find out the current balance.

## What Your CSRs Need to Know about BRM Customer Management

Customer Center includes online help for accomplishing tasks; however, you need to give your CSRs information that is specific to your business. For example:

- CSRs need to be able to tell your customers the following information about plans and deals:
  - Which services are included in each plan
  - The sign-up fee, monthly fees, and usage fees
  - Discounts, and when they apply
  - What happens when the plan is canceled; for example, is there a cancellation fee, and are fees paid in advance prorated and reimbursed
  - Expiration dates

- If CSRs can change service properties, they need to know what changes they are allowed to make; for example, if they can enter blocked numbers for a telephony service.
- CSRs need to know which information is required for registration. They also need to know the valid formats for entering information; for example, how to enter a telephone number, salutation, or title. For more information, see "Standard Registration Information" and "Specifying How to Validate Customer Contact Information".)
- Customers often have questions about changes to their account status. CSRs should know the reasons why an account can be inactivated and what needs to be done to activate an inactive account.

# About Maintaining an Audit Trail of BRM Activity

BRM provides support for auditing any object class 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.

BRM usually determines the fields that must be marked for auditing.

**Note:** You can also log CSR activities if you have included them in the **pin_notify** file. For more information, see "Logging Customer Service Representative Activities" in *BRM System Administrator's Guide*.

## About Using Audit Trails

Enabling an audit trail promotes good customer service. For example, customers can call your CSRs and find out when they changed the credit card number for their accounts. Assuming you are not encrypting credit card numbers, CSRs can also tell customers the credit card number they were using before the change. Another reason to enable an audit trail is to handle customer billing disputes. For example, a CSR can find which pricing plan a customer was signed up for during a certain billing period.

**Tip:** Enabling an audit trail decreases system performance significantly. Keep an audit trail only for the BRM activity you feel is absolutely necessary.

## About Accessing Audit Trails

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 the following topics in *BRM Developer's Guide*:

- "Accessing Audit Trail Information" for instructions on accessing audit trail information from the BRM database
- "About Tracking Changes to Object Fields" for more information on how audit trails work

- "Enabling Auditing for a Field" for instructions on making new and existing object fields auditable

# About Customer Center

Customer Center is a fully customizable Java application that CSRs use to manage interactions with your customers. Customer Center includes the following features:

- **Centralized installation**: You deploy Customer Center from a central Web server, making it available to each CSR to download and use locally. Each time a CSR starts Customer Center, the application checks the server and downloads any updates, using Java Web Start technology.
- **Customization**: You can add new panels to the Customer Center interface, make certain modifications to existing panels, and customize other aspects of Customer Center by using the Customer Center SDK.

## Who Uses Customer Center?

CSRs and accounts receivable (A/R) personnel are the primary Customer Center users. BRM implementers can also use Customer Center as part of testing BRM; for example, checking the balance impacts of price plans.

## Customizing Customer Center

One of the advantages of Customer Center and its modular framework is that you can customize it. You can customize Customer Center in the following ways:

- Rename or remove fields
- Change its basic look and feel
- Add functionality, such as entirely new fields or custom services (this type of customization requires programming)

For more information, see "Customizing the Customer Center Interface" in *BRM Developer's Guide*.

## About Using Customer Center

This section shows you the default tabs in Customer Center and explains how you work with account data on each tab.

---

**Note:** You can manage CSR access to Customer Center data and functionality. For more information, see "Setting Up Permissions in BRM Applications" in *BRM System Administrator's Guide*.

---

### Summary Tab

By default, this tab is active when you first open an account.

This tab gives you an overview of the open account, including the following:

- Name, address, and other contact information for the account holder
- Account summary, including the status, creation date, number of deferred actions, customer type, security codes, and if the account was created by the user
- Current balance

- Credit limit
- Number of unresolved disputes
- Payment method
- Current billing cycle dates

You can take the following actions by clicking links on the **Summary** tab:

- Change the account holder's information or create more contacts for the account
- Change the status of either the account or any service the account owns
- Display self-registration information, such as promotional codes, if your customer created the account on the Web
- Change the account's credit limit
- Resolve a dispute
- Change the account's payment method

### Balance Tab

This tab displays an overview of the account's balance and bills, including the following:

- Current balance for the account and for the bill in progress, or the bill for the current billing cycle
- Credit limit
- Summary of non-currency resources, if the account uses any
- List of the account's bills including the payment received for each bill

You can also do the following from this tab, by clicking links and using commands:

- Adjust the account; for example, to give the customer credit for a service problem
- Allocate adjustments or payments
- Display the bill in progress
- Change the account's credit limit
- Resolve a dispute
- View details of a bill
- View the invoice for a selected bill
- View A/R actions: adjustments, write-offs, and disputes, for all bills
- Search for and adjust events associated with a selected bill
- Write off a bill or an item
- Adjust an item for a selected bill
- Adjust an event
- Open and settle a dispute
- Refund a bill

### Payments Tab

This tab displays the following:

- A list of payments the account has made and how much of each payment has been allocated
- Payment setup information, which is different for each payment method
- Tax setup information

You can also do the following from this tab, by clicking links and using commands:

- View more details about a payment
- Search for and display payments for different time periods
- Allocate an unallocated payment
- Export the current list of payments to an HTML page
- Change billing information, including the payment method, name and address, and credit card number
- Add, delete, or change a tax exemption
- Export a list of payments to an HTML file

### Product Tab

This tab displays each product that the account owns and basic information about each product, including the product's service, purchase date, and status. You can click a product name to see more details, such as the product quantity, discounts, and fees, and to customize the product.

You can also do the following from this tab:

- Purchase new products
- Cancel a deal or product
- Customize the display of products on this tab
- Review the history of a product, deal, or service

### Service Tab

This tab displays the services that the account owns. For each service, it lists the login name or ID, status, and number of deferred actions. Some services may be subscription services or member services.

You can click a service's status to change it. You can change the status of a member service without affecting the other members of a subscription group. If you change the status of a subscription service, the status of its member services changes accordingly.

You can also change the login name or ID for a service and the password, if one is required.

---

**Note:** For some BRM services, additional fields appear below **New Password**.

---

#### View Alias

You can view the available alias or aliases for telco services. An alias is an additional identity associated with a service. For example, a service ID can have aliases such as International Mobile Subscriber Identity (IMSI) and Mobile Subscriber Integrated Services Digital Network Number (MSISDN) associated with it.

You use **View Alias** from the **Actions** menu to view the available alias or aliases for the selected telco service.

> **Note:** **View Aliases** is available only if the selected service is a telco service. If no aliases are available, the dialog box displays a message to that effect.

### Hierarchy Tab

For the parent account or member of an account hierarchy, you can see the accounts belonging to the hierarchy.

You can also do the following from this tab:

- Move any account into a hierarchy
- Move an account from one hierarchy to another
- Move an account to a different position within the same hierarchy
- Remove a child account from a hierarchy
- Open any account in the hierarchy

### Sponsorship Tab

In this tab, you can see whether an account owns a sponsor group or whether it is sponsored by another account.

For an account that owns a sponsor group, this tab displays the following:

- A list of groups that the account sponsors
- Products and rate plans for each sponsor group
- Members of each sponsor group

For a group owner, you can do the following from this tab:

- Create and delete sponsor groups
- Add or remove rate plans
- Add or delete group members
- Open any member's account

For an account that belongs to a sponsor group, you can display a list of sponsor groups and rate plans that the account belongs to. From that list, you can also display the account of the sponsor group's owner.

For any account, you can create a new sponsor group from this tab.

## About Event Browser

You use Event Browser to find events. When you find the event you are looking for, you can perform the following tasks:

- Summarize the impacts of events on the resource balances in an account. For example, you can see how login events during one month affected the balance of dollars and hours in a customer's account.
- Cancel the effects of events on resource balances. For example, if a customer was charged the wrong amount, you can cancel the charge by adjusting the event that generated the charge.

- Review detailed information about a customer's account such as credits, debits, dialup sessions, and name changes.

You access Event Browser from Customer Center. For more information, see "Using the Event Browser to Display and Adjust Events" in *BRM Managing Accounts Receivable*.

# About Dumping and Validating Account-Related Information

The Account Dump utility (ADU) is a diagnostics tool that allows 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.

You can use ADU to dump the contents of storable objects for an account into an output file and perform data analysis on the contents of the output file. For example, after an account migration, you can use ADU to dump the **/billinfo** and **/payinfo** object information of the account to a file and validate the billing and payment data.

**Important:** 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.

**Important:** 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 storable 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 1, 2007, and March 1, 2007. 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".

**Caution:** 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. See *BRM Developer's Reference*.

### Limiting Dump Information by Specifying a Date Range

You can limit the dump for the following storable 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 two types of validations on the object contents: structural and dynamic.

*Structural validation* validates the structure of the account. For example, validating that the POID of the parent bill object exists in the subordinate **/bill** object.

*Dynamic validation* validates the business logic. For example, validating that the **/item** due amount is zero after a payment.

Table 2–1 contains the predefined structural and dynamic validations. Each validation is associated with a validation code.

*Table 2–1 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 subordinate account's last bill points to the parent **/bill** object.<br>Validates that the bill number of the subordinate account and the bill number of the parent account are the same. |
| Structural | struct_ valid_03 | Validates that the DOM of the subordinate account and the DOM of the parent account are the same. |
| Structural | struct_ valid_04 | Validates that the AR_BILLINFO_OBJ of the subordinate account points to the **/billinfo** object of the parent account. |
| 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.

## 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 2–2:

*Table 2–2 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:<br>**- pin_adu input_file opt/portal/7.5/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:<br>**- pin_adu output_file /opt/portal/7.5/sys/diagnostics/pin_adu_validate/out** |
| **- pin_adu out_file_ext.***ext* | Set *ext* to the output file extension. For example:<br>**- pin_adu out_file_ext.xml** |
| **- pin_adu obj_list** *object1* **[***; object2***]**... | Specify the objects to dump for the selected accounts. For example:<br>To dump **/billinfo** and **/payinfo** objects:<br>**- pin_adu obj_list /billinfo; /payinfo**<br>**Note:** Use a semicolon to separate items in the object list. |

*Table 2–2 (Cont.) Entries in the ADU Configuration File*

| Entry | Description |
|---|---|
| **- pin_adu obj_flds** *object1*:*field1*, *field2*, ...<br>[*object2*:*field1*, *field2*, ...] ... | Specify the fields in the objects to dump. For example:<br>To dump the first and last name of an account:<br>**- pin_adu obj_flds /account: PIN_FLD_NAMEINFO.PIN_FLD_FIRST_NAME, PIN_FLD_NAMEINFO.PIN_FLD_LAST_NAME**<br>To dump account invoice information:<br>**- pin_adu obj_flds /payinfo/invoice: PIN_FLD_INV_INFO**<br>**Note:** Use a colon to separate the items in the object list. |
| **- pin_adu dump_start_time** *time*<br>**- pin_adu dump_end_time** *time* | Set *time* in these entries to the times to use for selecting most commonly updated objects. For example:<br>**- pin_adu dump_start_time 1175385600**<br>**- pin_adu dump_end_time 1177977600**<br>**Note:** *time* must be in UTC format. |
| **- pin_adu struct_valid_01** *n*<br>**- pin_adu struct_valid_02** *n*<br>**- pin_adu struct_valid_03** *n*<br>**- pin_adu struct_valid_04** *n*<br>**- pin_adu dynamic_valid_01** *n* | Use these entries to enable (**1**) or disable (**0**) the predefined validations. For example:<br>**- pin_adu struct_valid_01 1**<br>**- 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.<br>**- pin_mta logfile /opt/portal/7.5/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.<br>Log level values are as follows:<br>**0**: no logging<br>**1:** log error messages only<br>**2**: log error messages and warnings only<br>**3**: log error messages, warnings, and debug messages<br>**- pin_mta loglevel 2** |

# About Securing Sensitive Customer Data with Masking

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. Enabling data masking protects this data by hiding it.

## Enabling Data Masking

The **pin_fld_mask_file** file contains information about which data fields to mask.

To enable data masking:

1. Open the *BRM_Home*/**sys/cm/pin_fld_mask_file** file in a text editor.
2. For each field requiring masking, add the following line:

*Field_Name Length* [*Masking_Character*]

where:

- *Field_Name* is the field to be masked.
- *Length* is either the number of characters to mask or the number of characters at either end of the field to unmask.

  To specify the number of characters to mask, enter a numerical value. Masking a field with a set number of characters is useful when you do not want to reveal the length of a field.

  To unmask a set number of characters at either end of the field, specify the *Length* including a dash (**-**), the number of characters to unmask, and either an **L** or **R** to indicate whether to unmask the beginning (L) or end (R) of a field's string value.
- *Masking_Character* is the character used to mask the original value. If no masking character is specified, BRM uses * by default.

Example 2–1 shows a sample **pin_fld_mask_file** file.

***Example 2–1 Sample pin_fld_mask_file File***

```
# This file controls the masking of certain field values in flists the CM returns
# Only fields of type STR can be masked
# Each valid entry is of 1 of 3 types:
# 1.PIN_FLD_XXX   N    c #mask with a string of length N using char c
# 2.PIN_FLD_XXX   -NL  c #mask with char c leaving up to N chars unmasked on left
# 3.PIN_FLD_XXX   -NR  c #mask with char c leaving up to N chars unmasked on right
# In each case N is an integer >= 0 and c is the masking character
# (for type 1, N cannot exceed 128)
# The first type is useful for masking fields where we don't want to reveal how
# many characters make up the field.
# The remaining 2 types are useful for partial masking; note that the integer N #
# indicates how many characters are revealed.
#
# FIELD_NAME            LENGTH   MASK_CHAR
PIN_FLD_PASSWD          10           *
# mask with 10 '*'. Result: "**********"
PIN_FLD_PASSWD_CLEAR    15           -
# mask with 15 '-'. Result: "---------------"
PIN_FLD_DEBIT_EXP       -0L          0
# same as -0R, all positions are masked with 0
PIN_FLD_DEBIT_NUM       -4R          *
# the last 4 positions are not masked. e.g. "*******1234"
PIN_FLD_BANK_ACCOUNT_NO -2L          *
# the first 2 positions are not masked, e.g. "12**********"
```

3. Save the file.
4. Open the CM **pin.conf** file in a text editor.
5. Uncomment the following entry in the CM **pin.conf** configuration file:

   ```
   #- cm pin_fld_mask_file BRM_Home/sys/cm/pin_fld_mask_file
   ```

   **Note:** Add this entry in the CM **pin.conf** file if it does not already exist.

Uncommenting this entry enables masking of fields listed in the **pin_fld_mask_file** file.

6. Restart the CM. See "Starting and Stopping the BRM System" in BRM System Administrator's Guide.

**Note:** You must restart the CM whenever you change the masking configuration specified in **pin_fld_mask_file**.

## Masking Sensitive Data in Logs When Using Client Applications

Portal Communication Module (PCM) C++ and Java client applications may log flists containing sensitive customer data before calling the CM for processing and when returning results. Client application logs can contain flists in either 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.

# 3

# Customizing Registration

This chapter describes Oracle Communications Billing and Revenue Management (BRM) customer registration options that are common to customer service representative (CSR)–based and Web-based registration.

Before reading this document, see "About Registering Customers".

## Standard Registration Information

When you register customers, you collect information about them. The standard information includes:

- The plan that the customer signs up for.
- Contact information; for example, name, address, and telephone number.
- Service login and Web access names and passwords.
- Customer's language. See "Charging Customers at Registration".
- Customer's account currency.
- Billing information; for example:
  - The accounting type (balance forward or open item).
  - The frequency of the billing cycle.
  - The payment method (credit card, invoice, and so forth).
  - Payment information; for example, credit card number or invoice billing address.

The information required by default is the last name, address, city, and country. You can change the required fields by using the Field Validation Editor, a Configuration Center application.

If you require additional registration information, you can create custom fields. See "Creating New Registration Fields".

### Limitations for Entering Account Data

BRM uses default limits for the number of characters allowed for each field of customer data. For example, security codes can include up to 30 characters.

To find the number of allowed characters, see the description of the **/account** storable class. For example, the description of the PIN_FLD_ACCESS_CODE1 field includes this notation: Maximum length is 30 bytes.

### How BRM Uses the Customer's Language Setting

You can use the customer's language setting to deliver information to the customer in their language; for example:

- You can send localized versions of invoices to customers based on their language.
- Your Internet Protocol (IP) telephony gateway can play voice messages in the customer's language.

## Specifying How to Validate Customer Contact Information

Use the Field Validation Editor to define the valid entry formats for customer information. For example, when a customer or CSR enters a phone number, validation ensures that the phone number includes the correct number of digits.

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

**Tip:**

- You should tell your CSRs what the valid formats are.
- To see the default validation formats, see the Field Validation Editor in Configuration Center.

### Specifying the Valid Login Name Format

To specify the valid format for login names, use the Field Validation Editor. You should include a list of words that you do not want used as login names, for example, obscenities.

The default requirement is at least 1 character but no more than 255.

For more information, see "Managing Customer Authentication".

### Specifying the Valid Password Format

To specify the valid format for passwords, use the Field Validation Editor. You can create required combinations of alphabetic and numeric characters.

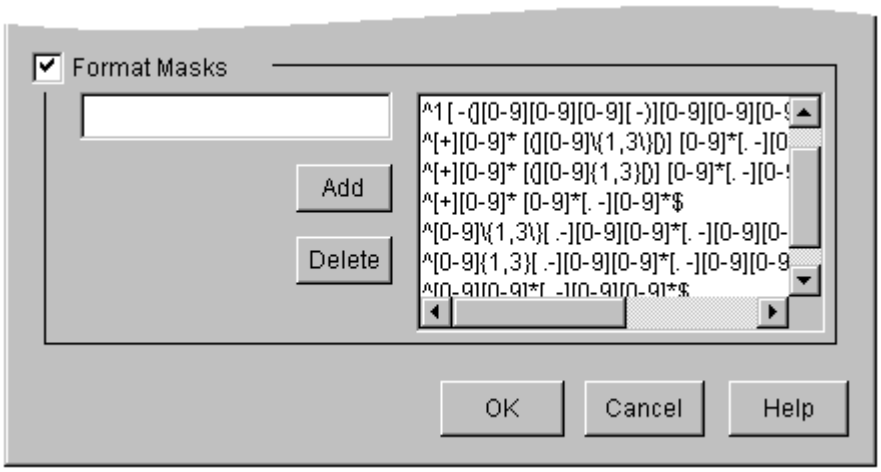
The default requirement is at least 1 character but no more than 255.

For more information, see "Managing Customer Authentication".

### Specifying the Valid Telephone Number Format

To specify the valid format for telephone numbers, use the Field Validation Editor. BRM includes several valid formats. To add a format, you create a format mask by using regular expressions. Figure 3–1 shows how format masks are entered in the Field Validation Editor.

*Figure 3–1 Format Masks*




## Specifying the US State Format

To specify the valid format for states, use the Field Validation Editor.

The default required format is two uppercase letters; for example, CA.

## Specifying the ZIP Code Format

To specify the valid format for ZIP codes, use the Field Validation Editor.

The default required format is 5 digits, or 5 digits followed by a hyphen and 4 digits.

**Note:** 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 the Salutation Format

To specify the valid format for salutations, such as Mr. or Ms., use the Field Validation Editor. If you include salutations on invoices, you might want to enforce the use of correct punctuation. If customers never see the salutation, punctuation is not as important.

**Tip:** To standardize account names, see "Standardizing Account Names".

## Specifying which Information is Required for Registration

You can specify which registration information is required. Some information, such as the customer's occupation, is usually optional.

By default, the following information is required:

- Last name
- Address
- City

- State
- ZIP code
- Country
- Service logins and passwords

> **Tip:** With Web-based registration, always let your customers know which information is required.

### Validating Data from Account Creation Applications

To validate account information during registration, use the PCM_OP_CUST_VALIDATE_CUSTOMER opcode. This opcode can validate partial information. For example, you can validate only part of the name and contact information.

Because the validation is performed during account creation and there is no account object yet, the input is a type-only Portal object ID (POID) instead of an account object. The input data consists of flist arrays containing zero or more of the following types of information:

- Name information
- Billing information
- Payment information
- Locale information
- Service login and password information
- Profile information

Each array has a flag to specify whether to validate partial information for that array. If the flag is set, PCM_OP_CUST_VALIDATE_CUSTOMER performs a partial validation. If the flag is not set, the opcode performs a complete validation.

If you provide a field that is dependent on another field for validation, you need to provide both the fields. For example, if the state, ZIP code, or phone field is provided for validation, then you must also provide the country field.

If there are no errors, PCM_OP_CUST_VALIDATE_CUSTOMER returns the results array in the output flist. If there are errors, it returns an flist with the first field that was not updated.

> **Important:** You need to explicitly delete the return flist after using it, by calling the PIN_FLIST_DESTROY routine.

## Credit Card Validation and Charge Options

By default, BRM validates a customer's credit card during account creation. You can set up the following credit card validation options:

- Specifying the Account That Records Credit Card Validations
- Allowing Registration without a Credit Card
- Validating Credit Cards at Registration
- Allowing Registration without Credit Card Validation

- Revalidating Credit Cards
- Charging Customers at Registration

## Specifying the Account That Records Credit Card Validations

By default, BRM logs credit card validations against the root account during registration.

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 you installed the BRM software.
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 "Customizing the Account Used for Credit Card Validation".

## Allowing Registration 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 register. 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 use Customer Center or a custom application to change the payment method to credit card or invoice after the free period has ended. (The custom application uses the PCM_OP_CUST_SET_BILLINFO opcode to set the payment method.)

## Validating Credit Cards at Registration

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 register 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 register 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.

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.

**Tip:** If you know that the connection will be down for a long time, you can edit the credit card processor Data Manager (DM) configuration file **online_proto** entry to avoid the delay caused by timeouts. See "Increasing Registration Speed When Paymentech Is Offline" in *BRM Configuring and Collecting Payments*.

**Important:** The PCM_OP_CUST_COMMIT_CUSTOMER opcode first calls the PCM_OP_PYMT_VALIDATE opcode to validate credit cards. If validation is successful, the customer account is created. Then, PCM_OP_CUST_COMMIT_CUSTOMER calls the PCM_OP_PYMT_COLLECT opcode to charge the credit card. If the authorization for funds fails, the account will *not* be charged.

## Allowing Registration without Credit Card Validation

By default, if there is no response from the credit card payment service, validation fails, resulting in registration failure because the credit card cannot be validated. To register customers without validating credit cards, do one of the following:

- If you use Paymentech, edit the Paymentech DM configuration file. See "Increasing Registration Speed When Paymentech Is Offline" in *BRM Configuring and Collecting Payments*.
- Modify the PCM_OP_PYMT_POL_VALIDATE policy opcode to map the “no answer” result to success, so that the registration continues.

**Tip:** If the credit card payment service is not available and you still want to register customers, you need to isolate those accounts for later credit card authorization. Modify the PCM_OP_PYMT_POL_VALIDATE policy opcode file either to save a list of permissive registrations or to send email to the system administrator. Alternatively, you can write a simple application to periodically check accounts and flag the ones that have been registered without verification.

## Revalidating Credit Cards

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 register for multiple accounts without having to validate the credit card more than once.

To specify the credit card revalidation interval:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the **cc_revalidation_interval** entry. The default is 3600 seconds (one hour).
3. Save the file.

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

## Charging Customers at Registration

By default, BRM charges for purchase fees and the first cycle forward fees at registration. When a customer registers, 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*.

> **Note:** 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 unapplied payments. For such accounts, Customer Center displays the following account balance information in the **Summary** tab:
>
> - Amount due for all bills: **0**
> - Adjustments/Payments not applied: **0**
> - Due now: **0**

To determine how to charge customers at registration:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the **-fm_pymt_pol cc_collect** entry:
   - To enable credit card collection, enter **1**.
   - To disable credit card collection, enter **0**.
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_COLLECT policy opcode. See "Customizing whether to Charge at Registration".

### Which Fees are Charged at Registration

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

### How General Ledger Revenue is Reported for Registration Charges

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

### Customizing whether to Charge at Registration

By default, BRM collects cycle forward and purchase fees on registration for credit card customers. Use the PCM_OP_PYMT_POL_SPEC_COLLECT policy opcode to customize whether to charge the customer immediately for all or part of the current account balances during registration or to defer the charges to a later date. You can also specify whether to validate the charges.

The only type of action that can trigger this opcode is **create customer**. If this is not passed in, PIN_ERR_BAD_VALUE is returned.

The PCM_OP_PYMT_POL_SPEC_COLLECT policy opcode checks all the account balance groups and specifies the amount for each bill unit.

- The PIN_FLD_BILLINFO_OBJ field in the input flist specifies the bill unit associated with the payment.
- The PIN_FLD_ITEMS array in the output flist specifies a list of open items to be paid.
- If the pay type is **_dd**, then it determines from the **pin.conf** file whether to validate, and then validates the account with Paymentech.

The PCM_OP_PYMT_POL_SPEC_COLLECT policy opcode then calls the PCM_OP_PYMT_GET_ACH_INFO opcode to retrieve the Automated Clearing House (ACH) information.

> **Note:** PCM_OP_PYMT_GET_ACH_INFO retrieves ACH information from the **/config/ach** object. It uses the ACH vendor name or element ID in the input flist to determine which element in the ACH_INFO array should be used.

BRM supports direct debit transactions from checking accounts only.

You can disable the default logic for the PCM_OP_PYMT_POL_SPEC_COLLECT policy opcode by changing the value of the **cc_collect** or **dd_collect** entry in the CM **pin.conf** configuration file. Change the value from **1** (enabled) to **0** (disabled). See "Charging Customers at Registration".

## Replacing Credit Card Numbers with Tokens at Registration

By default, credit card numbers entered during account registration are stored in the BRM database.

You can enable credit card tokenization in BRM to store tokens instead of actual credit card numbers in the BRM database. These tokens are then used for any BRM-initiated payments instead of the actual card numbers.

For more information on credit card tokenization, see the following in *BRM Configuring and Collecting Payments*:

- About Credit Card Tokenization
- Enabling Credit Card Tokenization

## Specifying Customer Account Defaults

You can set up system-wide defaults that are used by all registration methods. See the following topics:

## Specifying the Default Account Currency

You can specify the default account currency for new accounts. For more information, see "Managing System and Account Currencies".

To specify default account currency for new accounts:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Enter a currency code in the **currency** entry.

   For a list of currency codes, see *BRM_Home*/**include/pin_currency.h**.
3. Save 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.

## Specifying the Default Country

You can specify the default country for new accounts.

To specify the default country for new accounts:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the **country** entry:

```
- fm_cust_pol    country     USA
```

   **Note:** 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".
3. Save 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.

## Changing the Account Number Format

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 format of account numbers, customize the PCM_OP_CUST_POL_PREP_ACCTINFO policy opcode.

### Defining Customer Email Domain Names

You can change the email domain name that your customers use in their email addresses.

To change the email domain name:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the **domain** entry.

```
- fm_cust_pol   domain   isp.nz
```

3. Save 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.

### Specifying the Valid Expiration-Date Format

You can specify the valid date format for credit card and debit card expiration.

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 change the date format, customize the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode.

## Specifying the Maximum Number of Months Allowed in Billing Cycles

A billing cycle defines the time period between bills, which corresponds to the billing frequency (how often a bill is generated). For example, if the billing cycle is three months, a bill is generated every three months.

If a CSR specifies an account's billing frequency that exceeds the maximum number of months allowed in billing cycles, Customer Center returns an error and the CSR must change the billing frequency.

The default maximum number of months allowed in billing cycles is 24.

To change the maximum number of months allowed in billing cycles, use the Field Validation Editor to modify the value of the **bill_when** field.

## Allowing Accounts To Be Created with Backdated Services or Resources

By default, when you create an account, its associated services and resources 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 resource date backdated prior to the account creation date. To do so, enable the **SubsDis74BackDateValidations** parameter in the **subscription** instance of the **/config/business_params** object.

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

1. Go to *BRM_Home*/**sys/data/config**, which includes the support files used by the **pin_bus_params** utility.

2. Use the following command to create an editable XML file from the **business_params_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. To place this file in a different directory, specify the path as part of the file name.

3. Open the **bus_params_subscription.xml.out** file.

4. Search for the following line:

   ```
   <SubsDis74BackDateValidations>disabled</SubsDis74BackDateValidations>
   ```

5. Change **disabled** to **enabled**.

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

6. Save the file as **bus_params_subscription.xml**.

7. Use the following command to load this updated file into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

9. Stop and restart the CM.

# Controlling which Plans and Deals Are Available to Customers

To display plans and deals to CSRs in Customer Center, you use Pricing Center to include the plans in the following plan lists:

- Use the **CSR-new** plan list to display plans when you create new accounts in Customer Center.
- Use the **CSR-addon** plan list to display plans when you add a service to an existing account in Customer Center.
- Use the **default-new** and **default-addon** plan lists when creating accounts or adding services by using Web-based registration. In most cases, you will use your own plan lists.

> **Note:** The **CSR** plan lists override the **default** plan lists. If you have a **CSR** plan list and a **default** plan list, Customer Center displays only plans in the **CSR** plan list.

## Options for Presenting Plans and Deals

Displaying plans and deals in Customer Center is the easiest way to present information to customers because the CSR can describe each plan to the customer.

You can also present plans and deals based on customer input. For example, you can give customers a promotion code which, when entered on your Web page, displays a specific list of plans.

If you are creating a custom registration application, use the PCM_OP_CUST_POL_GET_DEALS and PCM_OP_CUST_POL_GET_PLANS policy opcodes to present plans and deals to customers.

### Modifying Deals During Registration

You can modify deals during registration; for example, you can provide discounts and price overrides. For more information on modifying deals, see "Managing Customers' Services and Products".

### Changing Plan and Deal Names and Descriptions

You create plan and deal names and descriptions when you create your price list in Pricing Center.

Plan and deal names and descriptions are displayed in Customer Center when creating an account and when adding services and deals.

## Creating New Registration Fields

Your business might require that you collect information about customers that is not part of the BRM standard set of registration information.

You can customize the Account Creation wizard in Customer Center to collect additional information by using Customer Center SDK. See "Using Customer Center SDK" in *BRM Developer's Guide*.

If you register customers by using a custom registration application or over the Web, you can create your own registration fields. To do so, you need to create new fields in Storable Class Editor. You can specify validation requirements for custom fields by using the Field Validation Editor.

You can also collect nonstandard information about customers by using profiles. See "About Collecting Nonstandard Account Information".

## Checking for Duplicate Registrations

You can have BRM check for duplicate customer registrations. For example, you can set up 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.

> **Note:** While preventing customers from accidentally signing up for a duplicate account, you also run the risk of preventing customers who want two accounts from signing up for the second one.

Checking for duplicate registrations requires editing the source code of the customer validation policy opcodes. See "Customer FM Policy Opcodes" in *BRM Developer's Reference*.

## Validating Registration Numbers

When registering customers, you can offer different plans to different customers by having customers enter registration numbers. You can also assign registration numbers to customers to track marketing information.

To specify how to validate registration numbers, edit the PCM_OP_CUST_POL_VALID_AACINFO policy opcode.

## Encrypting Customer Data

Developers can customize BRM to encrypt customer data such as credit card numbers, direct debit accounts, and so on. For more information on setting up encryption, see "Encrypting Fields" in *BRM Developer's Reference*.

## Exporting Registration Information to a Non-BRM Database

To export information to a non-BRM database, customize the PCM_OP_CUST_POL_POST_COMMIT policy opcode.

## Customizing Automatic Account Creation (AAC) Information

To prepare automatic account creation (AAC) data for validation, use the PCM_OP_CUST_POL_PREP_AACINFO policy opcode. This policy opcode takes the AAC fields for an **/account** or **/service** storable object during customer registration and processes them as necessary to prepare for validation.

To validate automatic account creation data, use the PCM_OP_CUST_POL_VALID_AACINFO policy opcode.

For more information, see "About the PREP and VALID Opcodes" in *BRM Developer's Reference*.

## Creating Hooks to External Programs

Use the following policy opcodes to implement customized registration functionality:

- PCM_OP_CUST_POL_PRE_COMMIT. See "Adding Custom Account Creation Steps before the Account Is Committed".
- PCM_OP_CUST_POL_POST_COMMIT. See "Adding Custom Account Creation Steps after the Account Is Committed".
- PCM_OP_CUST_POL_GET_CONFIG. See "Sending Account Information to Your Application when an Account is Created".
- PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER. See "Sending Account Information to Your Application when an Account is Modified".

### Adding Custom Account Creation Steps before the Account Is Committed

To run additional steps before the account is committed, use the PCM_OP_CUST_POL_PRE_COMMIT policy opcode.

The PCM_OP_CUST_POL_PRE_COMMIT policy opcode is called by the PCM_OP_CUST_COMMIT_CUSTOMER opcode just after the **/account** and **/service** storable objects have been created and initialized but before the transaction containing those operations has been committed. Therefore, the PCM_OP_CUST_POL_PRE_COMMIT

policy opcode cannot alter the **/account** and **/service** storable objects, but it can cancel the registration process by returning an **ebuf** error. Therefore, you can include tests that the customer must pass before proceeding.

The entire registration flist is passed in the input flist so any information about the customer can be used in interacting with the external or legacy system.

No information is returned as output parameters. Unless an error is returned, registration continues as expected.

The default implementation does nothing.

## Adding Custom Account Creation Steps after the Account Is Committed

To run additional steps after the account is committed, use the PCM_OP_CUST_POL_POST_COMMIT policy opcode.

The PCM_OP_CUST_POL_POST_COMMIT policy opcode is called by PCM_OP_CUST_CREATE_CUSTOMER just after the transaction containing the creation and initialization of the **/account** and **/service** storable objects has been committed.

The PCM_OP_CUST_POL_POST_COMMIT policy opcode takes the entire registration flist as its input flist so any information about the customer can be used to interact with an external or legacy system. This policy opcode cannot alter the registration data used to create the customer. Because the transaction creating the customer storable objects has already been committed, this operation cannot prevent or alter the registration process in any way. If an **ebuf** error is returned by the PCM_OP_CUST_POL_POST_COMMIT policy opcode, it is ignored by PCM_OP_CUST_CREATE_CUSTOMER.

No information is returned as output parameters.

### Sending a Welcome Email Message

The default implementation of the PCM_OP_CUST_POL_POST_COMMIT policy opcode supports sending a welcome email message to the new customer. See "Setting Up Introductory Messages".

## Sending Account Information to Your Application when an Account is Created

To send account information to your application after the account is created, use the PCM_OP_CUST_POL_GET_CONFIG policy opcode. This policy opcode is called by PCM_OP_CUST_COMMIT_CUSTOMER.

The PCM_OP_CUST_POL_GET_CONFIG policy opcode allows configuration information, such as news server address or mail server address, to be determined dynamically.

Input parameters include the POID of the account storable object that was created by the registration. Output parameters include all configuration fields and values that should be returned to the client software.

The default implementation supports different sets of configuration information based on the value of the **aac_source** field from the customer registration data. The list of configuration fields is stored in a text file in a configurable directory.

### Sending Account Information to Your Application when an Account is Modified

To send account information to your application after a service is purchased, use the PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER policy opcode. This opcode is called by the PCM_OP_CUST_MODIFY_CUSTOMER opcode.

The PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER policy opcode provides a mechanism to export customer data to an external or legacy system for processing when new services have been added to existing customers.

PCM_OP_CUST_MODIFY_CUSTOMER calls the PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER policy opcode just after the transaction containing the creation and initialization of the new **/service** storable object associated with the add-on plan has been committed. The input flist contains all of the same information as the input flist of PCM_OP_CUST_MODIFY_CUSTOMER, thereby allowing any information about the customer to be used by the external or legacy system. Because the transaction creating the customer **/service** storable object has already been committed, the PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER policy opcode cannot alter the customer data or prevent or alter the customer modification process in any way. If an error is returned by the PCM_OP_CUST_POL_POST_MODIFY_CUSTOMER policy opcode, it is ignored by PCM_OP_CUST_MODIFY_CUSTOMER.

## Returning a Point-of-Presence (POP) List

To get a registration point-of-presence (POP) list, use the PCM_OP_CUST_POL_GET_POPLIST policy opcode.

This opcode retrieves either the best POP for a registering customer to call or the entire list of POPs so the customer can choose one. Both approaches are supported because some registration models do not supply the customer phone number when retrieving the POP list, so no automatic matching can be done. If no phone number is supplied, the entire POP list is returned.

If the application requires the entire POP list, then all that is required on the input flist is the PIN_FLD_POID.

If the application is looking for the nearest POP based on location, the input flist requires the area code and prefix (or the ANI) in the PIN_FLD_ANI field and the PIN_FLD_POID field containing the database number where the POP tables reside.

The fields returned to the caller when a specific POP is needed are:

- PIN_FLD_PHONE
- PIN_FLD_CITY
- PIN_FLD_STATE
- PIN_FLD_ZIP
- PIN_FLD_FLAGS

The PIN_FLD_FLAGS field contains two flags:

- The POP_TOLL_FREE flag, if set, indicates whether the POP will be a toll free call for the caller.
- The POP_DIAL_ONE flag, if set, indicates that the caller needs to dial 1 first when placing the call to the pop.

These flags are bit flags. To check if POP_DIAL_ONE is set, for example, the code would be:

```
if (flags & POP_DIAL_ONE) {
/*
** Action code.
*/
}
```

The fields returned to the caller when a list of POPs is requested is an array of POPs (PIN_FLD_POP) where each element contains fields:

- PIN_FLD_PHONE
- PIN_FLD_CITY
- PIN_FLD_STATE
- PIN_FLD_ZIP

The default implementation uses **/pop** storable objects in the database to determine the POP information to return. Applications are provided with BRM to load POP storable objects into the database for each POP in the network and to compute the best POP for each exchange in the U.S. based on phone call rating tables from a third party.

# Creating Accounts in a Multischema System

In multischema systems, you can configure BRM to balance account loads among database schemas. See "Managing a Multischema System" in *BRM System Administrator's Guide*.

You can use policy opcodes to customize how to select schemas during account creation:

- To get a list of schemas, use the PCM_OP_CUST_POL_GET_DB_LIST policy opcode. See "Getting a List of Database Schemas".
- To select a schema, use the PCM_OP_CUST_POL_GET_DB_NO policy opcode. See "Selecting a Database Schema".

## Getting a List of Database Schemas

To get a list of database schemas defined in a multischema environment, use the PCM_OP_CUST_POL_GET_DB_LIST policy opcode. If you have schemas that you do not want included in the list of available schemas, you can customize this opcode to prevent them from being listed.

The default implementation returns the cached list of schema distributions created by the PCM_OP_CUST_POL_GET_DB_NO policy opcode during BRM initialization. It is designed for use by Brand Manager to enable you to select which schema should be used when creating a brand and then ensure the brand is created in that schema. This is crucial because all the sub-brands and accounts in the brand must be in the same schema as the top-level brand.

When BRM is set up as a multischema environment, the output flist contains a POID and the schema distributions array. In a single-schema environment, the output flist contains only the POID.

## Selecting a Database Schema

In a multischema system, the criteria for selecting a schema is set in the *BRM_Home*/**apps/multi_db/config_dist.conf** file. See "Setting Database Schema Priorities" in *BRM System Administrator's Guide*. During account creation, PCM_OP_CUST_

COMMIT_CUSTOMER calls the PCM_OP_CUST_POL_GET_DB_LIST policy opcode to select a schema for the new account.

You can customize the selection process by customizing the PCM_OP_CUST_POL_GET_DB_NO policy opcode.

If you are not using a multischema system, this step is ignored at account creation.

# 4

# Setting Up Customer Self Care with Self-Care Manager

This chapter describes 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.

For additional information about Self-Care Manager, see the following topics:

- For more information on Web registration and account access, see "Ways to Implement a Web Interface".
- 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 plans, 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 on 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.

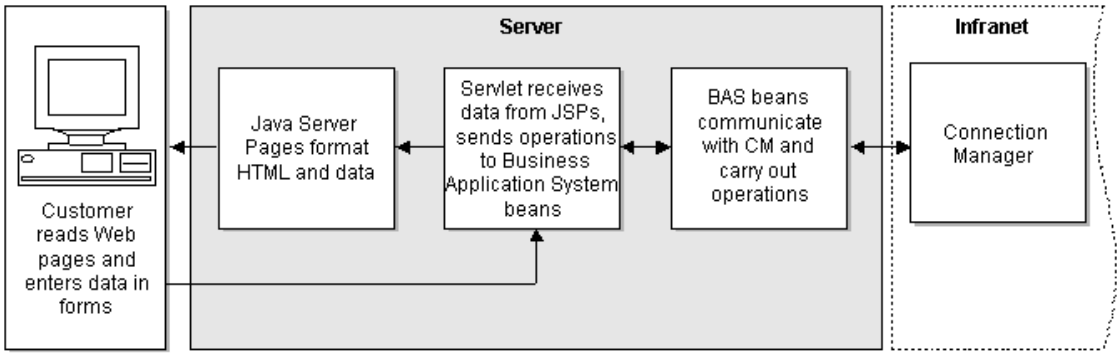
The JSPs and the application server are run by your Web server.

### Supported Application Servers

Self-Care Manager supports the Tomcat, WebLogic, and WebSphere application servers. For more information on supported versions, see "BRM Software

Compatibility" in *BRM Developer's Guide*. Figure 4–1 shows the flow of information.

***Figure 4–1 Self-Care Manager System Architecture***




## 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 register 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*.

# Installing Self-Care Manager

Install the software in this order:

1. BRM
2. Application server
3. Self-Care Manager

The application server and Self-Care Manager must be installed on the same system.

**Note:** The Connection Manager (CM) need not be on the same system as Self-Care Manager.

See "Installing Self-Care Manager on Windows" in *BRM Developer's Guide* or "Installing Self-Care Manager".

# 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".
- WebSphere. See "Setting Up IBM WebSphere".

## 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 you installed the BRM software.

### Deploying Your Application with Apache Tomcat

To deploy your Self-Care Manager application with Apache Tomcat:

1. Install Apache Tomcat. For more 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 more information, see the WebLogic documentation.
2. Create WebLogic domain. For more 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 more information, see the WebLogic documentation.
6. Stop and restart the WebLogic server.

   Your Self-Care Manager application is now deployed.

## Setting Up IBM WebSphere

This section explains how to deploy your Self-Care Manager application with IBM WebSphere.

### Requirements

- 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 *BRM_Home*/**WebKit/WebKit**.

### Deploying Your Application with IBM WebSphere

To deploy your Self-Care Manager application with WebSphere:

1. Install WebSphere. For more information, see the WebSphere documentation.
2. Deploy your **WAR** file by using the WebSphere application installation wizard.

   **Important:**

   - Even though you have a Self-Care Manager **WAR** file, deploy your application within WebSphere as an Enterprise Application (***.ear**) rather than as a **WAR** file.
   - Install your **WAR** file as a standalone application.

3. When prompted for enterprise application information, select the defaults, then click **Finish** to complete the wizard.
4. Copy the WebKit.properties and Infranet.properties files from *BRM_Home*/**WebKit/WebKit to** *Websphere_home*/**AppServer/installedApps/myapplication.**
5. Stop and restart the WebSphere server.

   Your Self-Care Manager application is now deployed.

## Uninstalling Self-Care Manager

To uninstall Self-Care Manager:

1. Uninstall Self-Care Manager from your application server. See your application server documentation.

   **Note:** You may also have to remove any temporary directories created for Self-Care Manager from your application server.

2. In Windows, do one of the following to start the uninstaller:
   - (Windows 8.1) Use the Windows uninstall feature.
   - (Other Windows versions) From the start options, choose **Uninstall Self-Care Manager**, which is under **Portal** or use the Windows uninstall feature.

## 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 more 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.**

**Caution:** If you use this method, you will encounter problems later if you try to use opcode and flist logging for troubleshooting.

For more 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, login type, and password.

**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 password is **password**.
   - The hostname is **host**.
   - The port is **40010**.

6. Save and close the file.

**Important:** To improve security, change the default root login name (**root.0.0.0.1**) and the default password (**password**). See "Configuring Login Names and Passwords for BRM Access" in *BRM System Administrator's Guide*.

## Specifying which Plan List to Display

By default, Self-Care Manager displays plans included in the **webclient** plan list. If the **webclient** plan list does not exist, Self-Care Manager displays plans included in the **default-new** plan list.

To specify a plan list for Self-Care Manager:

1. Open the Self-Care Manager properties file (*SelfCareManager_install_dir*/**WebKit.properties**).
2. Enter the name of the plan 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 deals 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
  ```

  **Note:** If your BRM license limits the number of BRM connections you can have, consider this limit when specifying the number of connections that Self-Care Manager can use.

- **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.

> **Important:** If you have multiple Self-Care Manager home pages, because your BRM system supports brands or for any other reason, 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
- Internet Explorer shows a "page not found" error

- Brands are enabled in BRM, but you want some accounts created in Brand Host

### 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
   ```

   **Important:** 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**.

### Internet Explorer shows a "page not found" error

Internet Explorer masks the actual error by giving the error message "page not found." You can turn off the error messages and see the actual error message by changing a setting in Internet Explorer:

To disable Internet Explorer error messages:

1. Choose **Tools - Internet Options**.
2. Click the **Advanced** tab.
3. Clear the **Show friendly HTTP error messages check box**.

#### Brands are enabled in BRM, but you want some accounts created in Brand Host

If your BRM system has branding enabled but no brands have been created, or you want to have accounts that do not belong to a specific brand, you need to set up **root** as a brand:

To set up **root** as a brand:

1. Open the Self-Care Manager properties file (*SelfCareManager_install_dir*/**WebKit.properties**).
2. Add the following line to the file:

   ```
   brand.root=admin_client,root.0.0.0.1,password
   ```

3. Open the **file htmlui_en/index.html.**

   **Locate the following line:**

   ```
   <a href="CreateFormPage1.jsp">Set up a new account</a></TD>
   ```

   Replace it with this line:

   ```
   <a href="CreateFormPage1.jsp?brand=root">Set up a new account</a></TD>
   ```

# 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 Web site. These JSPs serve HTML pages.

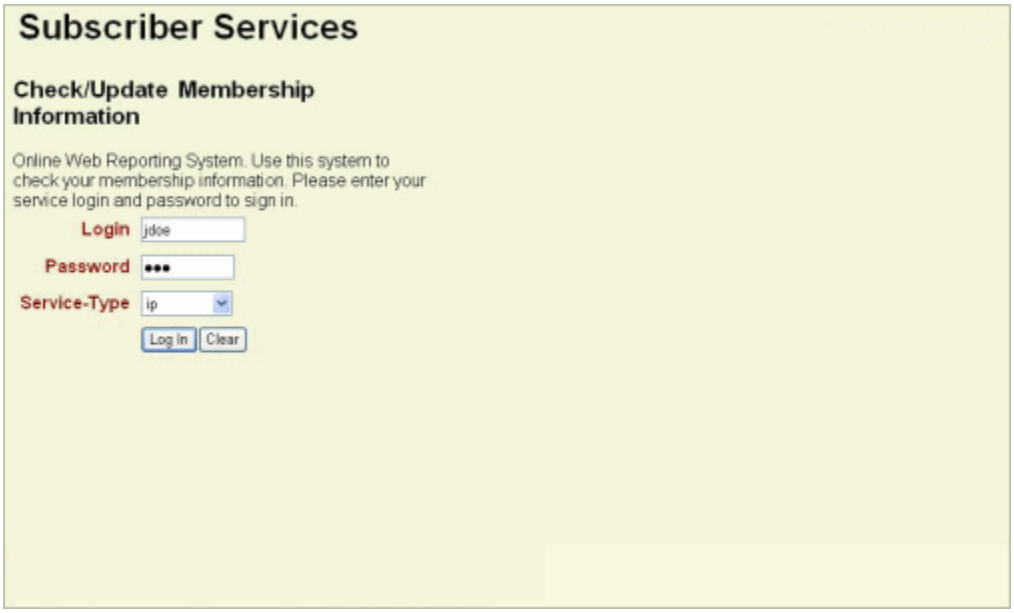
This section describes Self-Care Manager's default HTML pages and provides an overview of the functionality they offer.

For more information on customizing these pages, see "Changing Web pages" and "Customizing the Self-Care Manager Interface" in *BRM Developer's Guide*.

## Logging In

Figure 4–2 shows the login page your customers see when they access your self-care Web site:

*Figure 4–2 Login Page for Self-Care Web Site*




Self-Care Manager supports email, IP, and telephony service types.

## Accessing Account Information

After customers log in to your Web site on the login page, the Account Information page appears. This page displays the details of the account-level bill unit 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 as shown in Figure 4–3. For more information, see "Accessing Account Information".

***Figure 4–3 Example 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 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

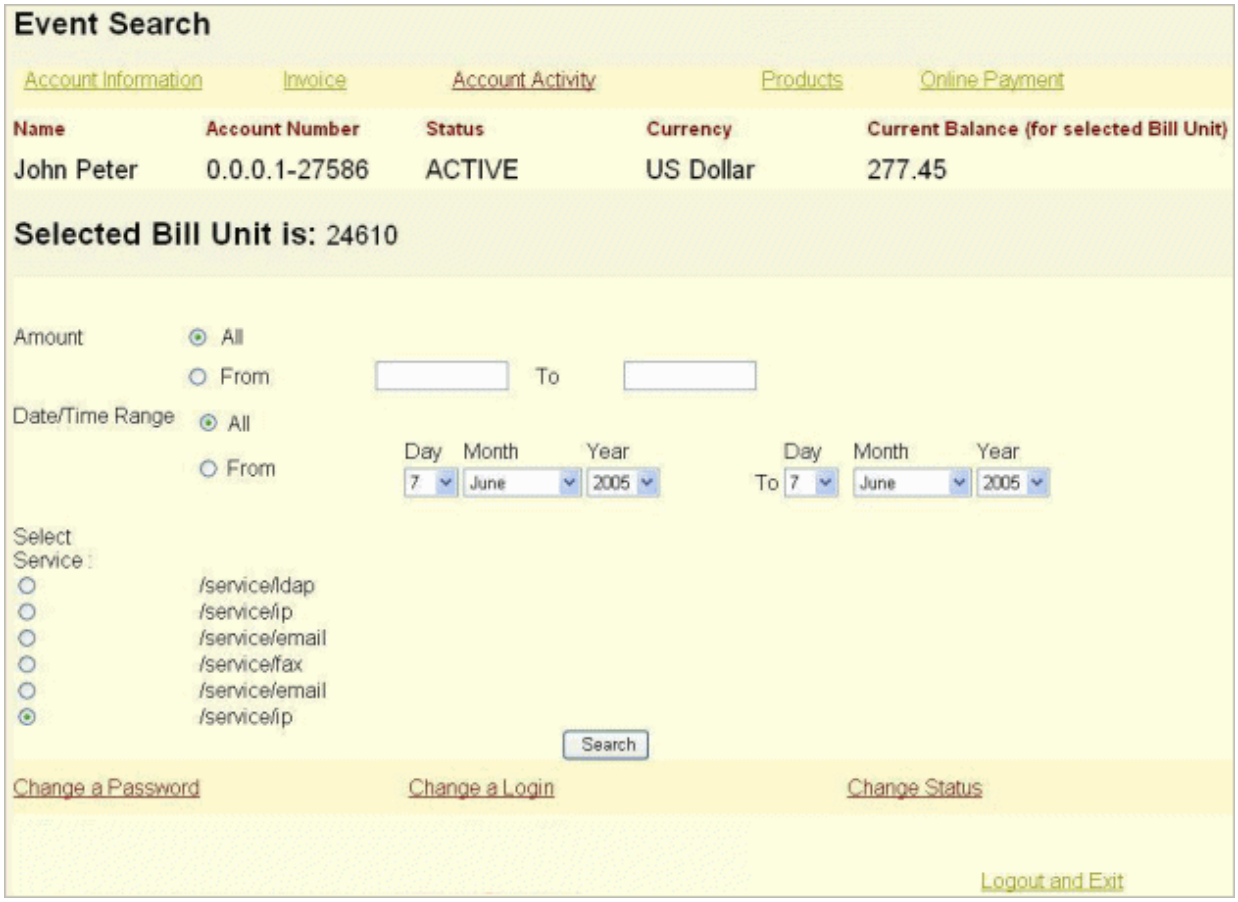
Customers can specify the search criteria and view the events of the selected bill unit. To enable this feature, you must configure the **item types** in the **WebKit.properties** file.

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 as shown in Figure 4–4.

***Figure 4–4 Example Event Search***




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 non-currency account balances. They can top-up one currency resource and an unlimited number of non-currency resources. For more information, see "About Standard Top-Ups" in *BRM Configuring and Collecting Payments*.

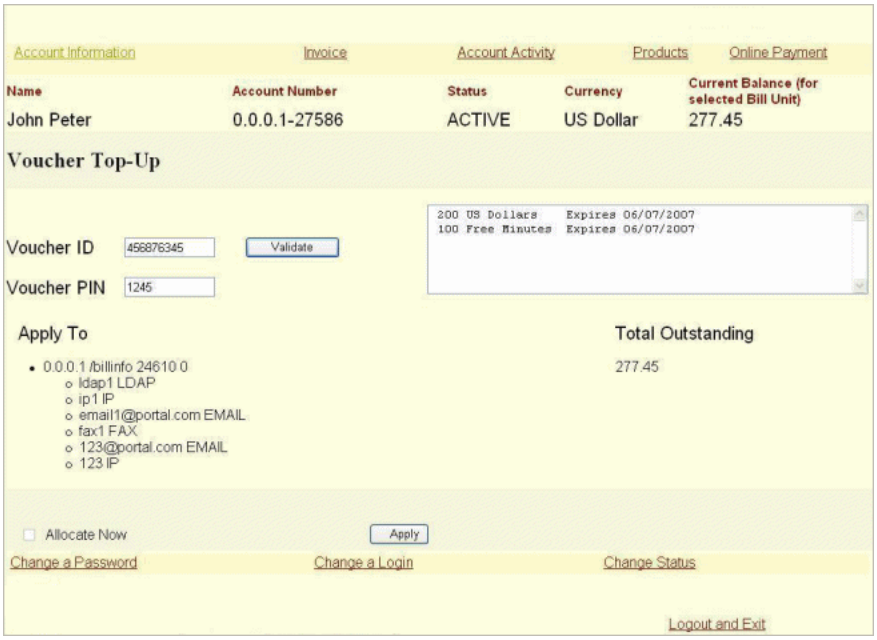
To top-up their account with a voucher, customers click the **Voucher Top-Up** link on the Account Information page. The Voucher Top-Up page appears. Customers can then enter the voucher ID and pin and click **Validate** to check the resources and validity of the specified voucher. If the voucher is 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**. See Figure 4–5.

Customers can also choose to just apply the voucher on 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.

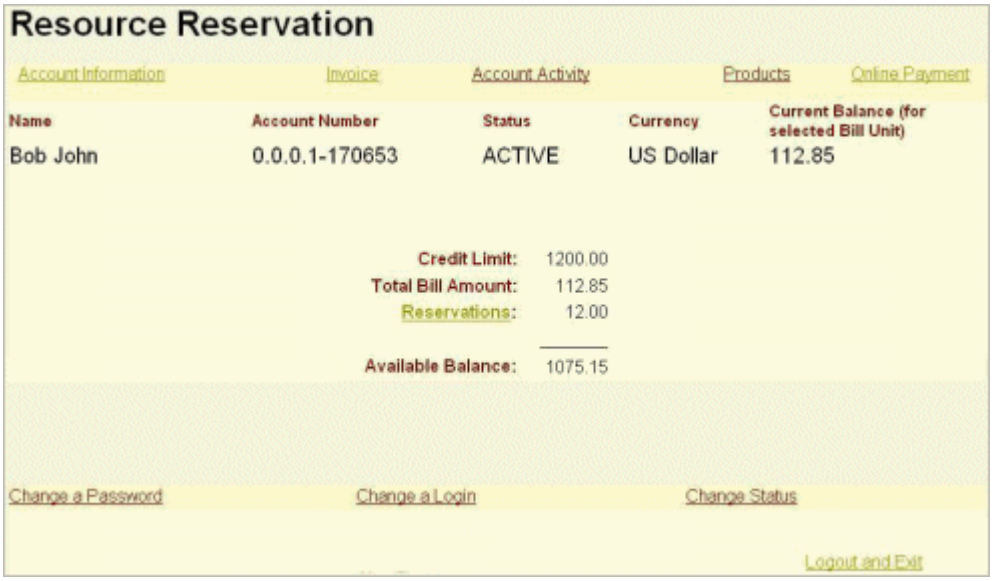
***Figure 4–5 Voucher Top-Up Information for an Example Account***




## Viewing Resource Reservation Details

Customers can view the resource reservation details of a bill unit if you have installed Resource Reservation Manager. To do this, customers click the **Resource Reservation** link on the Account Information page. The Resource Reservation page appears as shown below in Figure 4–6. It displays available credit limit, total bill amount, amount reserved, and available balance of all the balance groups in a bill unit.

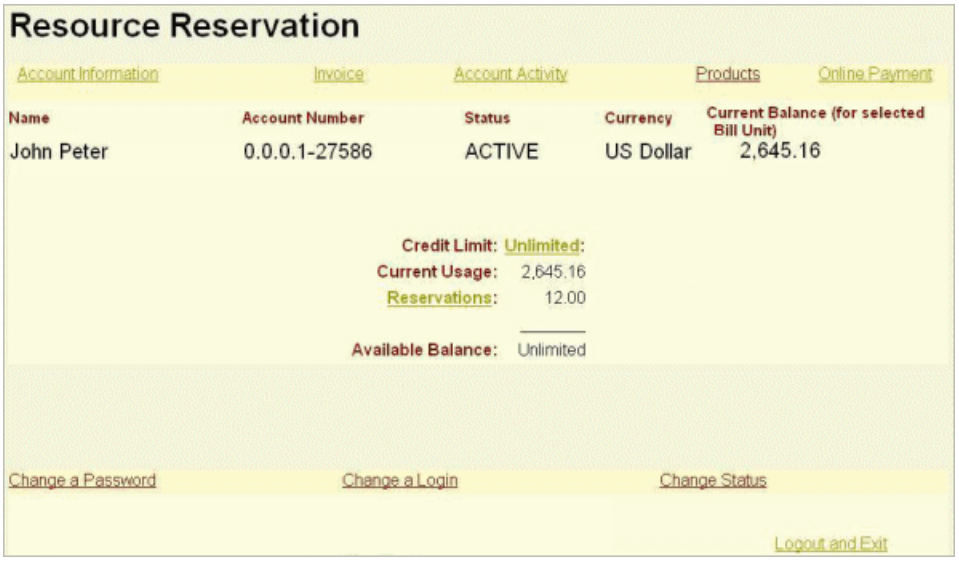
*Figure 4–6 Resource Reservation for a Selected Account*




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** as shown below in Figure 4–7.

*Figure 4–7 Resource Reservation with Unlimited Credit Limit*

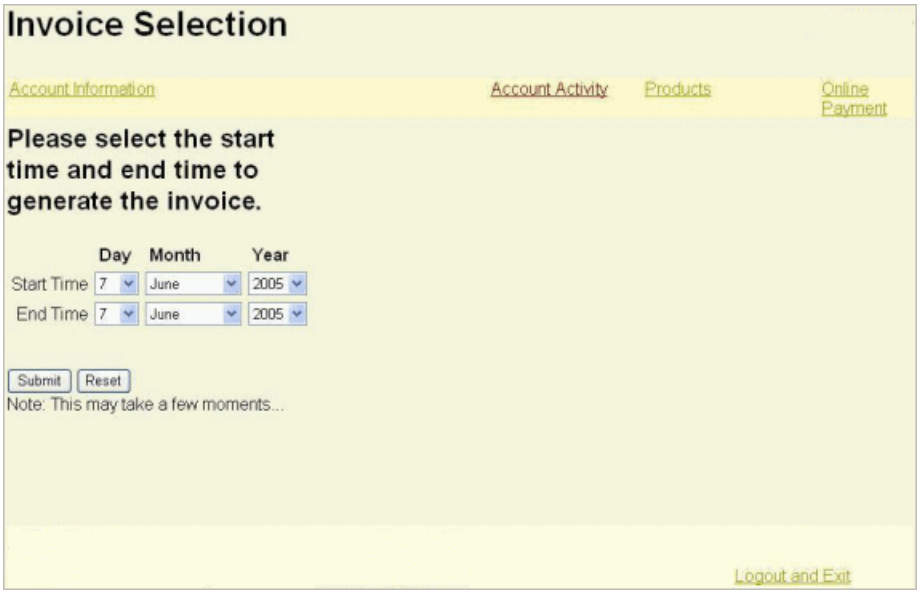



To find the balance group, which has unlimited credit balance customers, click **the Unlimited** link displayed next to the **Credit Limit**. The Included Services page appears. See "Viewing Service Details for a Bill Unit".

## 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 appears as shown in Figure 4–8.

*Figure 4–8 Selecting the Invoices to View*

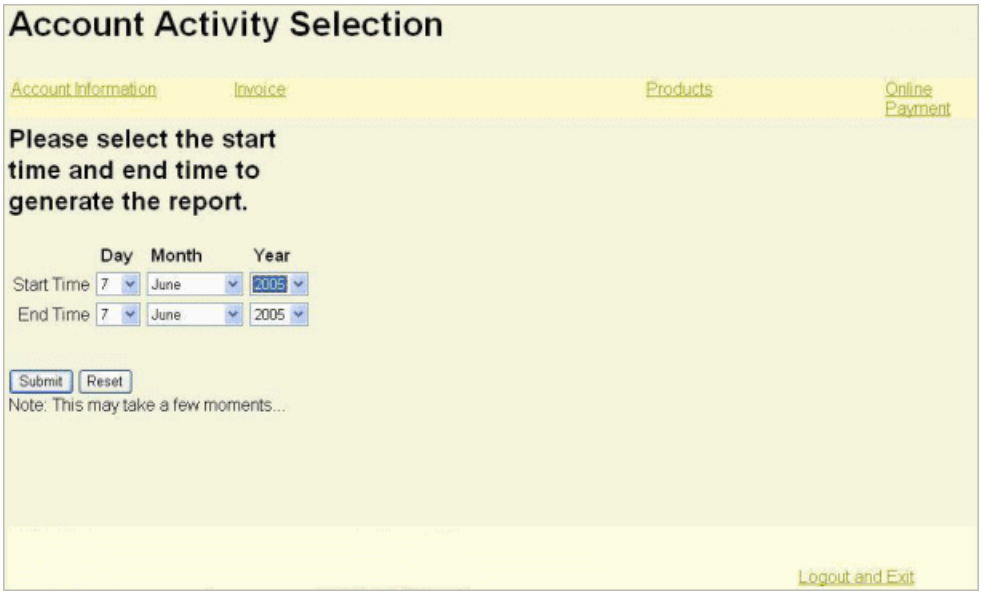



Customers can then select new start and end dates for the report or keep the default dates that represent the last billing cycle. After clicking **Submit**, the Invoice page appears, which lists invoices available for the selected period. In this page, customers click the invoice they want to view.

### 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. The Account Activity Selection page appears as shown in Figure 4–9:

*Figure 4–9 Account Activity Selection*




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 Purchased

To view the products and discounts purchased, customers click the **Products** link on the Account Information page. The Product Information page appears as shown in Figure 4–10:

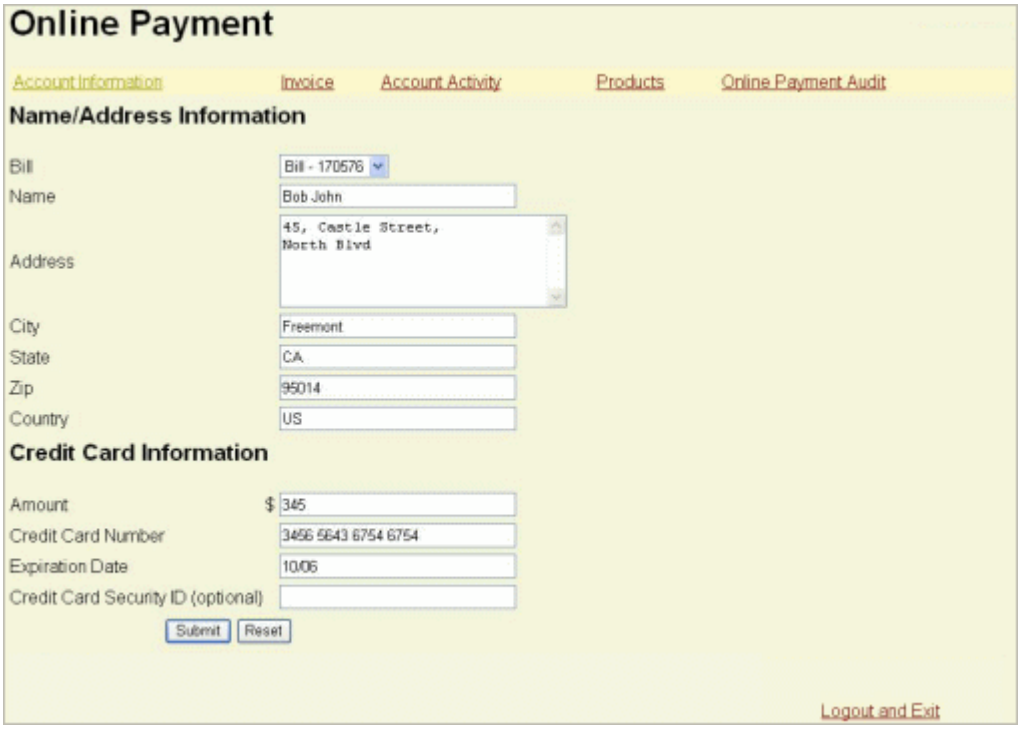
***Figure 4–10 Product Information Page***


If a discount was purchased as part of a deal, an icon appears before the deal name.

### Paying Bills Online

To pay bills online, customers click the **Online Payment** link on the Account Information page. The Online Payment page appears as shown below in Figure 4–11:

*Figure 4–11 Online Payment for Selected Account*




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.

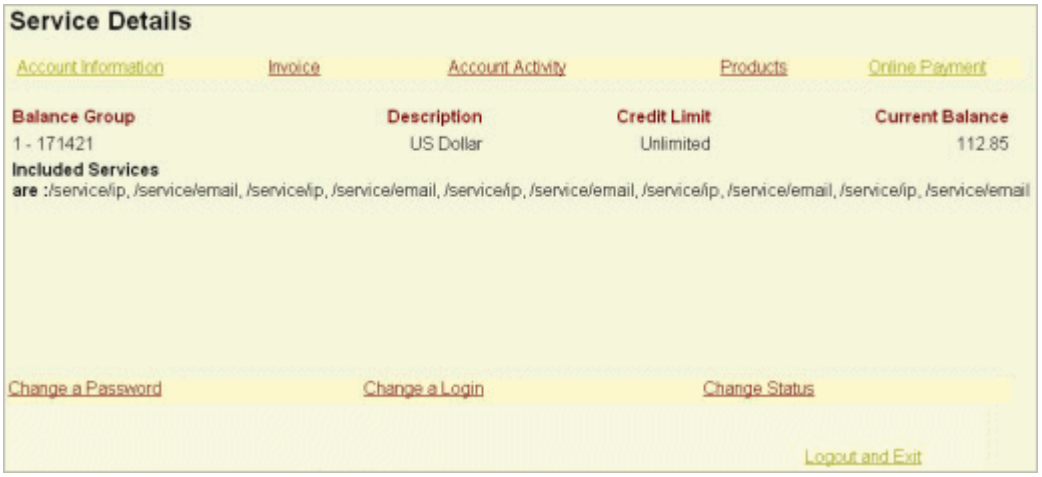
For more information, see "About Collecting BRM-Initiated Payments" in *BRM Configuring and Collecting Payments*.

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 (as shown in Figure 4–12) and displays the details of the services.

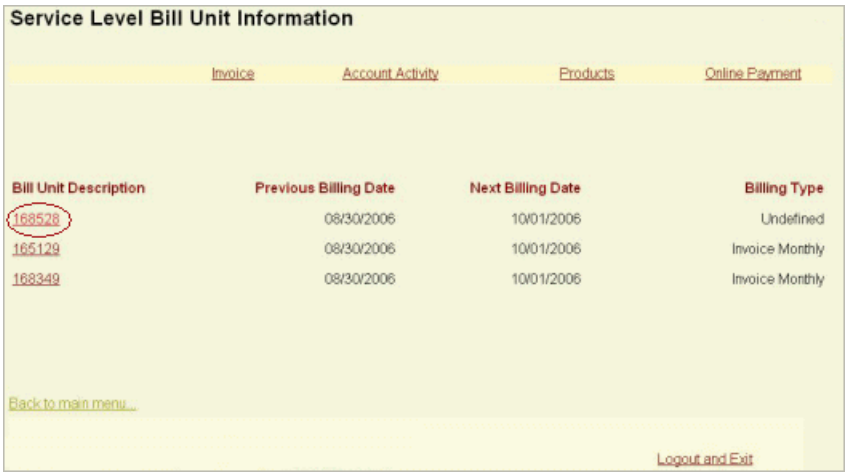
*Figure 4–12 Service Details for Selected Account*




## 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 as shown below in Figure 4–13.

*Figure 4–13 Service Level Bill Unit Information for Selected Account*




To view the details of a specific bill unit, click on the bill unit. The details are displayed in the Account Information page. For more information, see "Accessing Account Information".

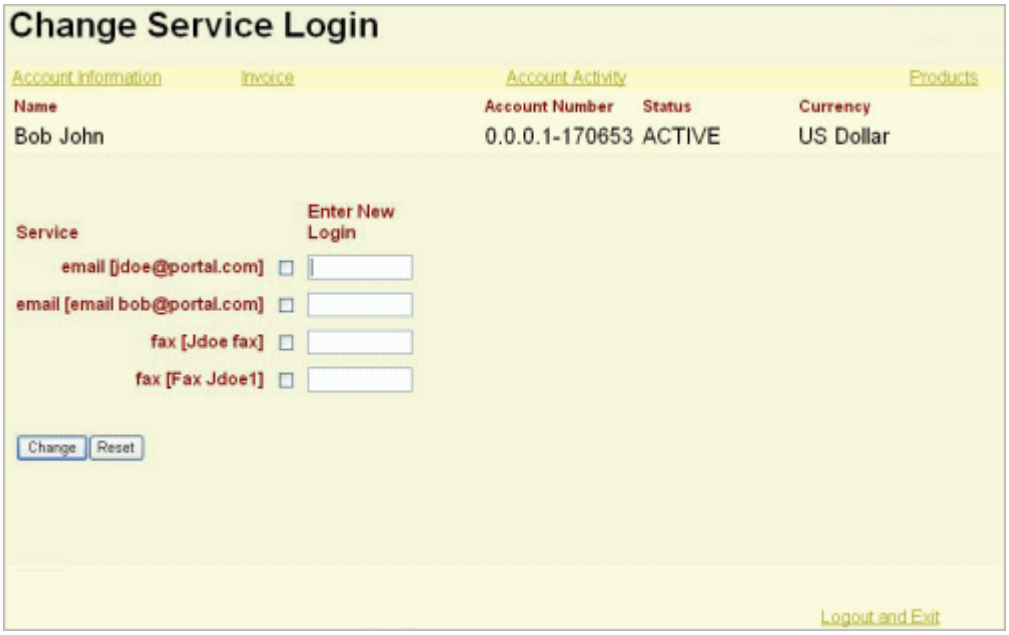
## Changing Login Name, Password, or Account Status

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**, pages similar to Figure 4–14 appear.

The account status can be active, inactive, or closed. For more information, see "About Activating, Inactivating, and Closing Accounts".

*Figure 4–14 Changing a Service Login*




# 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 more 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 more information on modifying these pages, see "Editing JSPs".

- **HTML pages**: The HTML registration 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.

---

**Important:** 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 "Using Customer Center SDK" and "Customizing the Self-Care Manager Interface" in *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.

**Important:**

- 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 more information that is required for registration. 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 On 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 "Using Customer Center SDK" in *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 more information on supported versions, see "BRM Software Compatibility" in *BRM Developer's Guide*.

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 4–1 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 4–1 Self-Care Manager JSPs***

| File Name | Description |
|---|---|
| **BrandSelect.jsp** | If the BRM server is brand-enabled, this JSP sets the scope to the correct brand, so that all information and data is specific to the subscriber's brand. |
| **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. |
| **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. |

*Table 4–1 (Cont.) Self-Care Manager JSPs*

| File Name | Description |
|---|---|
| **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 plans. |
| **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 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. |
| **AccountBalancePrepaid.jsp**<br>**AccountBalancePrepaidView.jsp**<br>**ReservationInfo.jsp**<br>**ReservationInfoView.jsp** | Display the resource reservation details of a bill unit. |
| **Event_details.jsp**<br>**Event_search.jsp**<br>**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 4–2 shows the HTML files used by Self-Care Manager:

*Table 4–2 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:<br><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 registration home page. Customers see this page first when they go to your Web site. |

## 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**.

# 5

# Sending Registration Information to Customers

This chapter describes how to automatically send messages to customers when they register and how to customize the messages in Oracle Communications Billing and Revenue Management (BRM).

## About Communicating with Your Customers

When your customers register for an account, whether over the Internet or through a customer service representative (CSR), you can configure your system to automatically send them messages during and after registration. Introductory messages might include:

- Information about your plans.
- The customer's connection settings, such as a new server address and mail server address.
- A list of phone numbers for dialup connections.

There are two ways you can send messages to your customers during online registration. You can use both methods:

- You can display a message in a Web page during Web registration. This message typically confirms the plan that the customer has selected.

  The default introductory message is located in *BRM_Home*/**sys/cm/intro/default.intro**. *BRM_Home* is the directory in which you installed the BRM software.

- You can send email after a customer has registered.

  The default email welcome message is located in *BRM_Home*/**sys/cm/welcome/default.welcome**.

## Sending Welcome Messages to Customers

If you want your customers to receive a welcome email message after they register, you need to enable the message. You also need to configure the Email Data Manager (DM) that sends the message. See "Sending Email to Customers Automatically".

### Enabling the Welcome Message

You must enable the welcome email message so that your customers can receive it after they register.

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

## Customizing the Welcome Message Text

Use any text editor to modify the text of the welcome email message. The default message is located in *BRM_Home*/**sys/cm/welcome/default.welcome**.

## Disabling the Welcome Message

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.

No email is sent and no error is reported.

## Using Variables to Insert Information into the Welcome Message

You can use variables to insert information based on customer input. For example, to display the plan that the customer selected, you use the ${price_plan} variable in the introductory message:

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

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

Table 5–1 shows the default variables:

***Table 5–1 Variables in Welcome Messages***

| Use This Variable | To Insert This Text | Example |
|---|---|---|
| **${price_plan}** | Plan name | Basic |
| **${plan_description}** | Plan description | Monthly Internet access |
| **${promo_code}** | Promotional code | Internet1 |

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

### Changing the Welcome Message 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 Message 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@your_domain.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:

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

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

### Specifying the Welcome Message 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.

## Setting Up Introductory Messages

The introductory message is an HTML file that you can display during Web-based registration, usually to confirm the customer's plan choice. 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. See "Specifying an Introductory Message".

### Customizing the Introductory Message

Use any text editor to edit the introductory message. You can add text and HTML tags. The default introductory message is located in *BRM_Home*/**sys/cm/intro/default.intro**.

## Using Variables to Insert Information into the Introductory Message

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

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

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

Table 5–2 shows the default variables:

***Table 5–2 Variables Used in Introductory Messages***

| Use This Variable | To Insert This Text | Example |
|---|---|---|
| **${price_plan}** | Plan name | Basic |
| **${plan_description}** | Plan description | Monthly Internet access |
| **${promo_code}** | Promotional code | Internet1 |

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

## Changing the Introductory Message Location

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.

## Specifying an Introductory Message

For more information on introductory messages, see "Sending Registration Information to Customers".

You can customize how to choose a message and how to send it:

- To choose the message, use the PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode.
- To send the message, use the PCM_OP_CUST_POL_POST_COMMIT policy opcode. See "Adding Custom Account Creation Steps after the Account Is Committed".

The PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode uses the type of account and plan to determine which message to send, so you can specify how to send each type of message.

The PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode reads the message from the file specified in the CM **pin.conf** file **intro_dir** entry. See "Changing the Introductory Message Location".

The PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode returns the introductory message as an uninterpreted buffer of data. This allows the introductory message to include HTML, graphics, and other complex information.

Within each introductory message, a parameter substitution system allows the message to contain values that are specific to this customer, such as the name of the plan they have chosen for purchase. For example, the following allows you to substitute a plan and a promotional code name:

```
You are requesting to purchase the ${price_plan} plan via the ${promo_code} promo
code.
```

The substitution is handled by the following code in the PCM_OP_CUST_POL_GET_INTRO_MSG policy opcode:

```
/*********************************************************
 * Price Plan Name
 *********************************************************/

if  (strcasecmp( macro, "price_plan")==0) {
valp = NULL;
valp = (char *)PIN_FLIST_FLD_GET( in_flistp,
PIN_FLD_AAC_PACKAGE, 0, ebufp);
if(valp) fm_cust_pol_buf_cat( bufp, valp, ebufp);
}

/*********************************************************
 * Promo code
 *********************************************************/

else if (strcasecmp( macro, "promo_code")==0) {
valp = NULL;
valp = (char *)PIN_FLIST_FLD_GET( in_flistp,
PIN_FLD_AAC_PROMO_CODE, 0, ebufp);
if(valp) fm_cust_pol_buf_cat( bufp, valp, ebufp);
}
```

### About the Email DM Opcodes

The PCM_OP_CUST_POL_POST_COMMIT policy opcode uses the Email DM PCM_OP_DELIVERY_MAIL_SENDMSGS opcode to send the message. PCM_OP_DELIVERY_MAIL_SENDMSGS in turn calls the Email DM PCM_OP_CREATE_OBJ opcode. See "Email Data Manager Opcodes" in *BRM Developer's Reference*.

## Using Multiple Welcome and Introductory Messages

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

Which message is displayed is based on a value entered during registration.

> **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

# 6

# Managing Customer Contact Information

This chapter describes how to use Oracle Communications Billing and Revenue Management (BRM) Customer Center to manage customer contact information, such as name, address, and telephone number.

## Managing Customer Contact Information

Use Customer Center to change customer contact information, such as name, address, and phone number.

When a customer service representative (CSR) or a customer changes customer contact information, BRM validates that it has been entered in the correct format. See "Specifying how to validate customer contact information".

**Note:**

- The **Credit Card Info** tab and the **Invoice Info** tab may contain duplicate information. If you edit information on either tab, ensure that you make the same changes to the other.
- If the customer has purchased an email service, BRM assigns an email address based on the login for the service. To change the email address, you must change the login. See "Changing Login Names".

### Specifying Multiple Account Contacts

You can store contact information for more than one person in a single account. For example, if an account is held by a large corporation, you can provide a different contact for each department. When you choose the contact, the contact's account information is displayed.

## Allowing Customers to Change Account Information

If you want customers to make their own changes at your Web site, create a Web page that allows customers to change their account information. See "Ways to Implement a Web Interface".

## Specifying an Invoice Contact

You can designate a person other than the account holder to receive the invoice. You can also use the same invoice contact for multiple accounts. See "Changing Invoice Information".

## Displaying Information Received from Web Registration

You can store information collected when customers register by using a registration kit or a Web site. If you collect this information, check the **Summary** tab in Customer Center to see the name of the third party where the customer registered and other information.

To collect this information, customize the PCM_OP_CUST_POL_PREP_AACINFO policy opcode.

## Managing Name and Address Information in Your Custom Application

For more information on validating customer contact information, see "Specifying how to validate customer contact information".

To add name and address information to a specific account object, use the PCM_OP_CUST_SET_NAMEINFO opcode. If the opcode updates the information in an existing account, it replaces the existing values with the new values. Any fields not specifically included in the input flist are left unchanged.

PCM_OP_CUST_SET_NAMEINFO sets the fields of the PIN_FLD_NAMEINFO array of a specified **/account** storable object to the values specified in the input flist PIN_FLD_NAMEINFO array. If PCM_OP_CUST_SET_NAMEINFO updates the information in an existing account, it replaces the existing values with the new values. Any fields not specifically included in the input flist are left unchanged.

PCM_OP_CUST_SET_NAMEINFO does the following:

- Sets the billing address in the account's PIN_FLD_NAMEINFO array by using the PIN_NAMEINFO_BILLING element (**element_id = PIN_NAMEINFO_BILLING = 1**).
- Sets the mailing address (if needed) using the PIN_NAMEINFO_MAILING element (**element_id = PIN_NAMEINFO_MAILING = 2**).

> **Important:** Element IDs through 100 are reserved for BRM's use.

The PCM_OP_CUST_CREATE_ACCT opcode calls PCM_OP_CUST_SET_NAMEINFO as part of the process of creating an **/account** storable object.

If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_CUST_SET_NAMEINFO creates an **/event/customer/nameinfo** storable object to record the details of the operation.

If the operation is successful, PCM_OP_CUST_CREATE_ACCT returns the BRM object ID (POID) of the **/event/customer/nameinfo** object in the PIN_FLD_RESULTS array. If PCM_OP_CUST_CREATE_ACCT fails, it returns a PIN_FLD_FIELDS array that specifies the failing field.

# Customizing Name and Address Information

Use the following policy opcodes to customize name and address information:

- PCM_OP_CUST_POL_PREP_NAMEINFO. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.
- PCM_OP_CUST_POL_VALID_NAMEINFO. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

PCM_OP_CUST_SET_NAMEINFO calls the PCM_OP_CUST_POL_PREP_NAMEINFO policy opcode to prepare the customer information for validation and to determine whether the account being created is the BRM payment suspense account.

The PCM_OP_CUST_POL_PREP_NAMEINFO policy opcode checks the relevant **/config/business_params** object to determine whether payment suspense management is enabled. If so, it checks if the first name is "payment" and the last name is "suspense," which identifies it as a payment suspense account. If it exists, the PCM_OP_CUST_POL_PREP_NAMEINFO policy opcode retrieves the POID of the **/config/psm** object and passes it to PCM_OP_CUST_SET_NAMEINFO.

If it is the payment suspense account, PCM_OP_CUST_SET_NAMEINFO performs the following tasks:

- If the policy opcode provides a **/config/psm** object, PCM_OP_CUST_SET_NAMEINFO updates the PIN_FLD_ACCOUNTS array with the account POID for the payment suspense account being created.
- If the policy opcode provides a dummy POID, PCM_OP_CUST_SET_NAMEINFO creates the **/config/psm** object. It includes the account POID of the payment suspense account in the PIN_FLD_ACCOUNTS_ARRAY.

PCM_OP_CUST_SET_NAMEINFO then calls the PCM_OP_CUST_POL_VALID_NAMEINFO policy opcode to validate the information.

## Creating Custom Country Aliases to Use in Client Applications

You can define custom values to represent countries by adding them to the country**:**alias list in the source code of the PCM_OP_CUST_POL_VALID_NAMEINFO policy opcode. This example shows three valid values for Angola: **AO**, **Angola**, and **Ang**.

```
"AO:AO",
"AO:Angola",
"AO:Ang"
```

After you edit this list, recompile the code to make your changes take effect.

For more information on the PCM_OP_CUST_POL_VALID_NAMEINFO policy opcode, see PCM_OP_CUST_POL_VALID_NAMEINFO and "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

# Specifying the Default Country

If the country is not provided, it is assumed to be USA and **USA** is added as the country value. You can edit the **country** parameter in the **pin.conf** file to change the default. See "Specifying the Default Country".

# Managing and Customizing Locale Information

Use the PCM_OP_CUST_SET_LOCALE opcode to set the locale in an account. This opcode creates an **/event/customer/locale** object (or one inherited from it) to record the details of the operation.

You can use the PCM_OPFLG_CALC_ONLY flag to run the operation without creating the event object.

To customize how locales are set, use these opcodes:

- To customize the creation of the locale, use the PCM_OP_CUST_POL_PREP_LOCALE policy opcode. The default implementation does nothing. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.
- To customize how to validate the locale, use the PCM_OP_CUST_POL_VALID_LOCALE policy opcode. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*. The PCM_OP_CUST_POL_VALID_LOCALE policy opcode does the following:
  - Validates limit information for a service.
  - Confirms that the value of PIN_FLD_LOCALE in the input flist is one of the valid BRM locales. See BRM locale IDs for a list of the valid BRM locales.

# About Collecting Nonstandard Account Information

You can customize BRM to collect information about your customers, in addition to the standard BRM account information. For example, you might want to collect the following information:

- How customers find out about your business.
- Services that your customers would like you to offer.
- Account numbers from legacy customer management systems.
- Information necessary to offer some services. For example, you might need to assign nicknames for a chat service.

The information you collect is stored in account profiles. You can use information in profiles in various ways; for example:

- You can create applications that search for information in profiles. For example, you could create a "friends-referral" program by creating a profile that collects referrals. When a customer registers, you can search profiles to find which customer made the referral and give that customer a credit.
- You can read information from an account profile and use it as input to custom applications and configurations. For example, you could base custom rates on values in a profile.

## Managing and Customizing Profiles

For more information on profiles, see "About Collecting Nonstandard Account Information".

- To add a **/profile** object to an account, use the PCM_OP_CUST_CREATE_PROFILE opcode.

  This opcode calls other opcodes that prepare and validate the account data. After the account data is validated, PCM_OP_CUST_CREATE_PROFILE calls base opcodes to create a **/profile** storable object that contains extra information that you

want to store about an account and associates it with the account storable object POID.

- To modify a profile object, use the PCM_OP_CUST_MODIFY_PROFILE opcode.

  Given the POID of an existing **/profile** object, PCM_OP_CUST_MODIFY_PROFILE modifies the object with the specified changes.

  If the input flist for the inherited fields for the profile object contains a null array or substruct value, the array or substruct table entry is deleted in the database.

- To delete profiles, use the PCM_OP_CUST_DELETE_PROFILE opcode.

To customize profiles, use the following policy opcodes:

- Use the PCM_OP_CUST_POL_PREP_PROFILE policy opcode to add custom data to the **/profile** object. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.
- Use the PCM_OP_CUST_POL_VALID_PROFILE policy opcode to validate the data in the **/profile** object. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

To display profile information in Customer Center, you must configure Customer Center by using Customer Center SDK. See "Using Customer Center SDK" in *BRM Developer's Guide*.

## Searching for Account Profile Information

You can modify Customer Center to perform searches on information in profiles.

To find the **/profile** objects that belong to an account, use the PCM_OP_CUST_FIND_PROFILE opcode.

This opcode uses the input PIN_FLD_POID database field of the input flist to determine which database to search for the profile list. The opcode calls a standard opcode to search for the profile.

If the PCM_OPFLG_READ_RESULT flag is set, PCM_OP_CUST_FIND_PROFILE returns the entire **/profile** storable object. Otherwise, it returns the fields specified in the PIN_FLD_RESULTS array. If no fields are specified, only the profile POID is returned. Other fields are returned with the profile POID only if they are specified in the PIN_FLD_PARAMETERS field of the input flist.

To limit the returned profile list, include a profile type string in the input flist by using PIN_FLD_TYPE_STR. A type string contains an object name or a wildcard value using the percent sign (%) that is compared to the profile object name. For example, **/profile/foo** returns all object subtypes "foo," and **/profile/f%** returns all profile subtypes starting with "f." If the profile string is not included in the input flist, all matching profiles for the account are returned.

**Note:** The percent sign is the only wildcard you can use with this field.

# 7

# Managing Customer Authentication

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

## About Customer Authentication and Authorization

*Authentication* verifies a customer's identity. By default, this is accomplished by checking the customer's login name and password.

*Authorization* verifies that a customer is allowed to use the service. A user might not be authorized if the service is inactive or if a credit limit has been reached.

Authorization requires that authentication occur first, so that BRM knows who is being authorized.

To customize authentication, you need to edit the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode. For example, you can change the requirements for authentication. See "Authenticating Customers by Using Your Custom Application".

You use the **pin_ipass_loader** utility to customize authorization for IP services.

## 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.
- A login name can be assigned to only one account. Login names are unique.

All login names and passwords are associated with a service; for example, the IP service. Customer service representatives (CSRs) use a login name and password to log in to the **admin_client** service.

> **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.

For more information, see "Setting Up Login Name and Password Defaults".

## About Encrypting Customer Passwords

BRM uses two types of customer passwords:

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

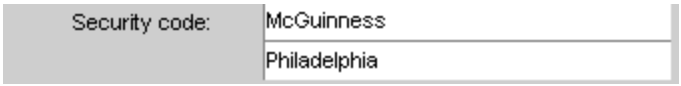
By default, account passwords are stored in the database in an encrypted format; service passwords are not.

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

## About Customer Security Codes

You can specify two security codes for a customer. When a customer calls a CSR, the CSR asks the customer for the security code, which is displayed in Customer Center as shown below in Figure 7–1:

***Figure 7–1 Customer Security Code***




Customers can change their security codes at any time. Security codes are not validated by BRM. See "Using Customer Security Codes".

# Assigning New Login Names and Passwords

By default, login names and passwords are assigned at account creation or when adding a service.

In Customer Center, you assign new login names and passwords in the "Service Tab".

# Changing Login Names

Use Customer Center to change login names.

**Important:** Changing the customer's login name for an email service also changes the customer's email address for the service. Before changing a login name, make sure the customer wants to change the email address.

# Changing Passwords

Use Customer Center to change passwords.

If you want customers to change their own passwords, create a Web page that enables customers to change their account information. See "Ways to Implement a Web Interface".

Typically, a customer's password must be changed in the following places:

- A file on the customer's computer
- The BRM database, by using Customer Center

To change the password on the customer's computer, have the customer locate their password file and make the change. The name and location of the password file differ depending on the configuration of the customer's connection software. For example, if your customer uses Netscape, the password file is usually located in the **netscape\dialer**\ISP**.sr** file, where ISP is the name of your company. However, because customers can change the default location of their password file when they install their connection software, they might have trouble finding this file.

### Replacing Lost Passwords

A customer password is not displayed in Customer Center. If a customer loses a password, try the following:

- Change the password in Customer Center on the **Service** tab. Have this customer reconfigure the dial-in software with the new password.

  You can also set up BRM to charge a fee for changing the password. See "About Charging for Administrative Events" in *BRM Setting Up Pricing and Rating*.

- If the customer cannot change the password in the dial-in software, have the customer contact the company that produces the dial-in software to find out whether there is a way to locate the lost password.

- If the customer cannot find or change the password, close the current BRM account and open a new one. This may be the easiest method for many customers.

  The drawback to this method is that closed accounts remain in your company's database. See "Reusing Login Names and Passwords from Closed Accounts or Canceled Services".

## Using Customer Security Codes

You create customer security codes in Customer Center. When a customer calls a CSR, the CSR asks the customer for the security code to authenticate the customer.

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.

Because security codes require a CSR to validate them, they are not used for Web-based administration. To control access to Web pages, use standard Web password methods.

You can change security codes at any time by using Customer Center.

### Assigning New Security Codes

Security codes are assigned at account creation. In Customer Center, you assign security codes on the **Contact Information** panel.

Use Customer Center to change and assign security codes.

## Setting Up Login Name and Password Defaults

You can change the minimum and maximum login name and password lengths by using the Field Validation Editor. You can also customize logins and passwords as follows:

- Assigning passwords automatically. See "Assigning Passwords Automatically".

- Standardizing how account names are displayed, regardless of how they are entered. For example, you can remove spaces from the name or capitalize the first and last name. See "Standardizing Account Names".
- Defining the required case-sensitivity of email logins. See "Defining Email Login Names".
- Detecting duplicate logins. See "Detecting Duplicate Logins".
- Defining how to encrypt passwords. "About Encrypting Customer Passwords".

### Assigning Passwords Automatically

You can set up registration 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.

### Standardizing Account Names

You can standardize how account names are displayed, regardless of how they are entered. 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 registration. To change this option, customize the PCM_OP_CUST_POL_PREP_NAMEINFO policy opcode.

### Defining Email Login Names

To change the requirements for case-sensitivity in email login names, customize the PCM_OP_CUST_POL_PREP_LOGIN policy opcode.

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: john_smith@oracle.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, customize the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode.

## Customizing Login Names

For more information on how login names are used in BRM, see "About Login Names and Passwords".

To set the login for an account, use the PCM_OP_CUST_SET_LOGIN opcode. This opcode sets the login field for a **/service** storable object to the value specified in the PIN_FLD_LOGINS array of the input flist.

If the PCM_OPFLG_READ_RESULT flag is set and the operation is successful, PCM_OP_CUST_SET_LOGIN returns all fields of the **/event/customer/login** storable object in the PIN_FLD_RESULTS array. Otherwise, only the Portal object ID (POID) of the event object is returned. If the operation is not successful, PCM_OP_CUST_SET_LOGIN returns the PIN_FLD_FIELDS array which specifies the failing field.

If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_CUST_SET_LOGIN creates an **/event/customer/login** storable object to record the details of the operation.

## Customizing Login Names

Use the following opcodes to customize how login names are created. These opcodes are called by PCM_OP_CUST_SET_LOGIN.

- To prepare login names, use the PCM_OP_CUST_POL_PREP_LOGIN policy opcode. For example, you can add characters or change the case to all lowercase. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.
- To validate login names, use the PCM_OP_CUST_POL_VALID_LOGIN policy opcode; for example, to ensure that they have the required number of characters, See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

  You define login validation rules by using the Field Validation Editor. This application loads validation rules into the **/config/fld_validate** storable object. By default, login names must be 255 characters or fewer.

**Caution:** BRM requires logins for services (they cannot be NULL). BRM requires that login names be unique. BRM will not function properly if the PCM_OP_CUST_POL_VALID_LOGIN policy opcode is customized to allow non-unique login names.

## Requiring Login Names for Email and IP Services

By default, BRM requires all accounts that purchase email or IP services to create a login name and password before they can proceed with the account creation or modification process. BRM enforces the login name requirement by using the following two policy opcodes:

- The PCM_OP_CUST_POL_PREP_LOGIN policy opcode converts all NULL login names to an empty string. For example, if the service type is **/service/email** and the policy opcode's PIN_FLD_LOGIN input flist field is set to NULL, the policy opcode changes the PIN_FLD_LOGIN flist field to **""**.
- The PCM_OP_CUST_POL_VALID_LOGIN policy opcode rejects all login names that are set to an empty string, because the validation process requires that all login names are at least one character in length.

To override this behavior and allow accounts to purchase email and IP services without supplying a login name, customize the PCM_OP_CUST_POL_PREP_LOGIN policy opcode to skip the conversion of NULL login names to an empty string.

## Creating Logins for Prepaid Services

For prepaid services, the login is generated automatically. The default login generated by the PCM_OP_CUST_POL_PREP_LOGIN policy opcode is a unique string composed of the database number, service name, and POID.

In cases where a prepaid service login is needed, such as Self-Care Manager, the customer's Mobile Station Integrated Services Digital Network (MSISDN) number is stored in the service object PIN_FLD_ALIAS_LIST array. See "About Using BRM for Wireless Services" in *BRM Telco Integration*.

The PCM_OP_CUST_POL_VALID_LOGIN policy opcode enforces that a password for a service cannot be changed when the account is created or when a service is added. The password can be changed later. See "About Telco Service Logins and Passwords" in *BRM Telco Integration*.

### Creating Logins for Email Services

For email services, the PCM_OP_CUST_POL_PREP_LOGIN policy opcode appends the **domain** entry in the Connection Manager (CM) configuration file (**pin.conf**) to the login. Passing in a domain name results in an error.

## Creating Passwords

For more information on how passwords are used in BRM, see "About Login Names and Passwords".

To add or change passwords, use the PCM_OP_CUST_SET_PASSWD opcode.

This opcode sets the PIN_FLD_PASSWD field for a specified **/service** storable object to the value specified in the PIN_FLD_PASSWORDS array of the input flist.

If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_CUST_SET_PASSWD creates an **/event/customer/password** storable object to record the details of the operation.

Before setting the new password value, PCM_OP_CUST_SET_PASSWD calls the following policy opcodes to perform these operations:

- PCM_OP_CUST_POL_PREP_PASSWD, to process and prepare the new password for validation.
- PCM_OP_CUST_POL_VALID_PASSWD, to validate the password.
- PCM_OP_CUST_POL_ENCRYPT_PASSWD, to encrypt the password.

If the PCM_OPFLG_CALC_ONLY flag is not set, PCM_OP_CUST_SET_PASSWD creates an **/event/customer/password** storable object to record the details of the operation.

If the password is successfully updated, the POID of the **/event/customer/password** object is returned in the PIN_FLD_RESULTS array. Otherwise, PCM_OP_CUST_SET_PASSWD returns the PIN_FLD_FIELDS array that specifies the failing fields.

### Customizing Passwords

Use the following policy opcodes to customize passwords:

- To prepare account or service passwords for validation, use the PCM_OP_CUST_POL_PREP_PASSWD policy opcode. This policy opcode takes the password field for an **/account** or **/service** storable object during customer registration and prepares it for validation.

  If the password is for an account, it is set to **NULL**.

If a service password is passed in, the operation does nothing. If there is no password for a service, it generates one. A password is considered to be passed in if the PIN_FLD_PASSWD_CLEAR field is in the input flist, even if it is NULL.

- To validate account or service passwords, use the PCM_OP_CUST_POL_VALID_PASSWD policy opcode. The default check is to make sure the password is not NULL and is less than 255 characters.

## Implementing Password Encryption

Use the following policy opcodes to implement password encryption:

The PCM_OP_CUST_POL_ENCRYPT_PASSWD policy opcode encrypts a cleartext password based on the type of the account or service POID given and/or the requested encryption algorithm. The binary result is stored as an ASCII-like string to facilitate storage. All encryption requests from IP accounts get clear text encryption (to support Challenge Handshake Authentication Protocol, or CHAP). Advanced Encryption Standard (AES) is used for all others. For more information, see "About AES Encryption" in *BRM Developer's Guide*.

- The PCM_OP_CUST_POL_COMPARE_PASSWD policy opcode takes a cleartext password and an encrypted password from the PCM_OP_CUST_POL_ENCRYPT_PASSWD policy opcode and performs a comparison to check if the cleartext password was the source value of the encrypted password. The type of the **/account** or **/service** storable object whose password is being compared is included in the input flist, so the encryption mechanism can be varied for different storable object types. The PCM_OP_CUST_POL_COMPARE_PASSWD policy opcode returns true (match) or false (no match). The PCM_OP_CUST_POL_COMPARE_PASSWD policy opcode is used by BRM whenever a client application supplies a password to be authenticated against an **/account** or **/service** storable object.
- The PCM_OP_CUST_POL_DECRYPT_PASSWD policy opcode decrypts a cleartext password.

## Creating Passwords for Prepaid Services

For prepaid services, the password is generated automatically. the PCM_OP_CUST_POL_PREP_PASSWD policy opcode generates a default password (**password**), which can be customized. See "About Telco Service Logins and Passwords" in *BRM Telco Integration*.

## Customizing Password Expiration

To customize password expiration, use the PCM_OP_CUST_POL_EXPIRATION_PASSWD policy opcode.

This policy opcode calculates and sets the expiration date for the password. This policy opcode is called when the password status of a CSR account is set as **Expires**.

By default, the PCM_OP_CUST_POL_EXPIRATION_PASSWD policy opcode sets the password expiration date to 90 days.

To change the default password expiry duration, edit the **passwd_age** entry in the CM **pin.conf** file. For example, instead of 90 days you can set the expiration duration to 150 days. See "Setting the Default Password Expiry Duration" in *BRM Developer's Guide*.

If successful, the PCM_OP_CUST_POL_EXPIRATION_PASSWD policy opcode returns:

- PIN_FLD_POID: The POID of the **/account** object passed in.
- PIN_FLD_PASSWORD_EXPIRATION_T: The expiration date and time when the password expires.

For more information, see "Managing CSR Passwords" in *BRM Developer's Guide*.

# Tracking Customer Authentication and Authorization Records

You can record authentication failures to detect possible service problems or the occurrence of fraud. For example, duplicate logins might indicate fraudulent usage. You can specify which authentication failures to record.

You can record user authentication or authorization 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.

## Specifying and Loading Verification Preferences

To record user authentication and authorization 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 "Creating Configuration Files for 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 you installed the BRM software. 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.

   > **Caution:** 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) Use Customer Center to 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 plan that includes no rates and has a purchase level of all accounts and no services.
   - 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 in *BRM_Home***/sys/cm**.
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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## Viewing Login Failures

To view login failures:

1. Use Customer Center to open the root account or the account you specified in the CM configuration file.
2. Open Event Browser.
3. Search for verification activity events.

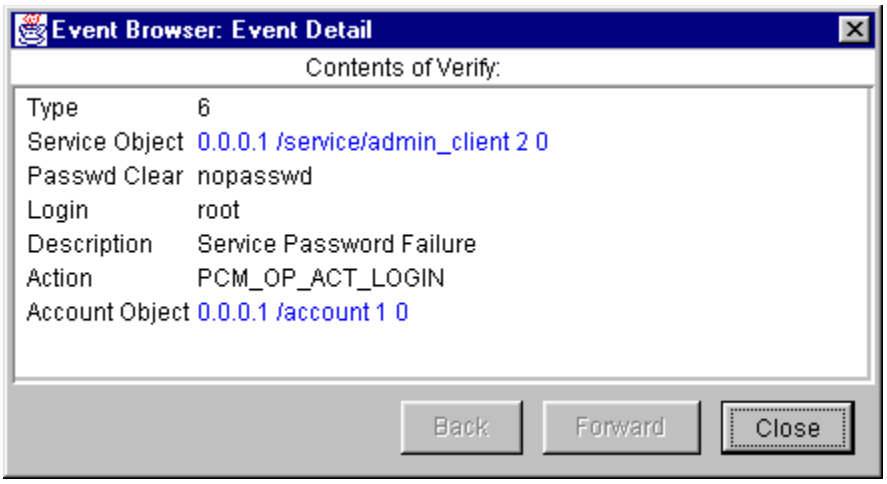
   Following are two examples of how to do this:

   - If you have an account set up especially to log verification events, you can search for all events logged to that account.

- If you are using **root** or another account that is logging many types of events, you can use the Any Event (Advanced) template in the Search dialog box to search only for events of the type **/event/activity/verify**.

Figure 7–2 is an example of what a verification event looks like in Event Browser:

***Figure 7–2 Verification Event in Event Browser***




See "About Event Browser" for more information.

# Authenticating Customers by Using Your Custom Application

When customers login, BRM first verifies the customer's login name and password and then performs a variety of checks; for example, when customers attempt to access a service or content offered by your company or a third-party provider.

- PCM_OP_ACT_FIND_VERIFY is the recommended opcode for authenticating customers. See "Authenticating User Actions".
- PCM_OP_ACT_FIND. See "Finding a Customer's Account Information".
- The PCM_OP_ACT_POL_SPEC_VERIFY policy opcode. See "Customizing Authentication Checks".

## Authenticating User Actions

Use PCM_OP_ACT_FIND_VERIFY to verify a customer's identity and to perform authentication checks, such as verifying that the customer has good credit and an active account status.

PCM_OP_ACT_FIND_VERIFY takes as input a type-only POID, the action to verify, and the user's login and password. PCM_OP_ACT_FIND_VERIFY then:

1. Calls PCM_OP_ACT_FIND to retrieve the user's **/account** and **/service** objects. See "Finding a Customer's Account Information".
2. Calls the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode to retrieve the list of authentication checks for the specified action. See "Customizing Authentication Checks".
3. Performs all authentication checks specified by the policy opcode.
4. Returns the following, depending on the success of the transaction:

- If authentication succeeds, PCM_OP_ACT_FIND_VERIFY returns the POID of the service object and the PIN_FLD_RESULT field set to one of the following:
  - **0** to indicate a valid login and password.
  - **4** to indicate that the customer used a temporary password.
- If authentication fails, PCM_OP_ACT_FIND_VERIFY returns PIN_FLD_RESULT set to one of the following:
  - **2** to indicate an incorrect password.
  - **5** to indicate that the customer used an expired password.
  - **6** to indicate that the password is no longer valid.

## Finding a Customer's Account Information

Use PCM_OP_ACT_FIND to locate a customer's account information by using the login name.

PCM_OP_ACT_FIND searches for information on a specific account. In addition, the opcode performs the following operations:

- If you use a multischema BRM system, it searches all database schemas to find the **/account** object.
- If there is an open context (CM connection) to a schema when it is called and it finds the account in a different schema, it switches the context to the correct schema.
- If there is no open context when it is called, it searches all schemas.

**Note:** PCM_OP_ACT_FIND does not perform authentication.

PCM_OP_ACT_FIND searches for the **/account** and **/service** object. If PCM_OP_ACT_FIND is called with the PCM_OPFLG_READ_RESULT flag, it returns all fields of the **/service** storable object, not just the POID.

## Customizing Authentication Checks

Use the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode to specify the list of authentication checks to perform for each user action.

This policy opcode is called by other opcodes to specify a list of checks used to authenticate user actions.The specified checks are then performed by standard opcodes.

Table 7–1 lists the available authentication checks:

***Table 7–1 Default Authentication Checks***

| Authentication Check | Enum Value | Description |
|---|---|---|
| PIN_ACT_CHECK_UNDEFINED | 0 | None |
| PIN_ACT_CHECK_ACCT_TYPE | 1 | Check type of **/account** storable object. |
| PIN_ACT_CHECK_SRVC_TYPE | 4 | Check type of **/service** storable object. |
| PIN_ACT_CHECK_SRVC_STATUS | 5 | Check the service status. Can be **active**, **inactive**, or **closed**. |

*Table 7–1 (Cont.) Default Authentication Checks*

| Authentication Check | Enum Value | Description |
|---|---|---|
| PIN_ACT_CHECK_SRVC_PASSWD | 6 | Confirm that the password is correct. |
| PIN_ACT_CHECK_CREDIT_AVAIL | 7 | Check the available credit. |
| PIN_ACT_CHECK_DUPE_SESSION | 8 | Check the open session count for the service. See "Enabling Duplicate Session Checking". |

You can control the list of checks that each action requires and, to some extent, the behavior of the checks. Few authentication checks lend themselves to these behavior changes. Those that do can define a PIN_FLD_CHOICES array for the options.

The other fields present in an element of the checks array depend on which of the checks is specified by that element. See the output flist specification for more details.

Table 7–2 shows the authentication checks that the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode returns and their behavior:

*Table 7–2 Authentications Checks Returned by PCM_OP_ACT_POL_SPEC_VERIFY*

| Action | Default Authentication Checks |
|---|---|
| PCM_OP_MAIL_DELIV_VERIFY | PIN_ACT_CHECK_SRVC_STATUS = active |
| PCM_OP_ACT_LOGIN | PIN_ACT_CHECK_SRVC_STATUS = active<br>PIN_ACT_CHECK_SRVC_PASSWD<br>PIN_ACT_CHECK_CREDIT_AVAIL = >=0 |
| PCM_OP_MAIL_LOGIN_VERIFY | PIN_ACT_CHECK_SRVC_STATUS = active<br>PIN_ACT_CHECK_SRVC_PASSWD<br>PIN_ACT_CHECK_CREDIT_AVAIL = >=0 |
| PCM_OP_TERM_IP_DIALUP_AUTHORIZE | PIN_ACT_CHECK_SRVC_STATUS = active<br>PIN_ACT_CHECK_SRVC_PASSWD<br>PIN_ACT_CHECK_CREDIT_AVAIL = >=0 |
| PCM_OP_TERM_IP_DIALUP_AUTHENTICATE/CHAP<br>PCM_OP_TERM_IP_DIALUP_AUTHENTICATE | PIN_ACT_CHECK_SRVC_STATUS = active |
| DEFAULT (for everything else) | PIN_ACT_CHECK_SRVC_STATUS = active<br>PIN_ACT_CHECK_SRVC_PASSWD |

You can customize the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode to:

- Choose a different combination of authentication checks to use on an action.
- Change the behavior of the authentication checks.
- Decide not to use any checks to authenticate an action.
- Authorize a custom action using the list of authentication checks.

The PCM_OP_ACT_POL_SPEC_VERIFY policy opcode uses the DEFAULT set of authentication checks on any action it does not recognize.

If you want to specify a different set of authentication checks for a new action, you must:

1. Edit the *BRM_Home*/**source/sys/fm_policy/fm_act_pol/fm_act_pol_spec_vrfy.c** source file.
2. Create a new Activity Policy Facilities Module (FM). See "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.

   Each authentication check must have a **type** (the enum value from the table of authentication checks above), and *choice* options to control the behavior of the check. See the PCM_OP_MAIL_DELIV_VERIFY action definition in the **fm_act_pol_spec_vrfy.c** file for an example of a PIN_FLD_CHOICES array.

For example, suppose you have defined a non-currency resource. If you want to enforce a credit limit on that resource, you must modify the **fm_act_pol_spec_vrfy.c** file to include a new action that checks the non-currency resource balance. (By default, checks are performed only on the currency resource.)

Reducing the number of authentication checks does not have a significant effect on performance, with the exception of duplicate session checking.

### Enabling Duplicate Session Checking

You can customize the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode to check for duplicate login attempts (PIN_ACT_CHECK_DUPE_SESSION) by performing the following tasks:

1. Modify your *BRM_Home*/**source/sys/fm_act/pol/fm_act_pol_spec_vrfy.c** file to add duplicate session checking to the input flist for the desired actions. This is just the **term_ip_dialup** related action. Assuming the default version of **fm_act_pol_spec_vrfy.c** is used as a base, add the following code fragment at line 339:

   ```
   type = PIN_ACT_CHECK_DUPE_SESSION;
         c_flistp = PIN_FLIST_ELEM_ADD(a_flistp, PIN_FLD_CHECKS, 4, ebufp);
         PIN_FLIST_FLD_SET(c_flistp, PIN_FLD_TYPE, (void *)&type, ebufp);
         action = "/dialup";
         PIN_FLIST_FLD_SET(c_flistp, PIN_FLD_OBJ_TYPE, (void *)action, ebufp);
   ```

2. Add the same code fragment at line 390.
3. Recompile the file and use it as a new Activity Policy FM.

# 8

# Creating and Managing Customer Segments

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) customer segment feature and provides information on how to create customer segments.

For more information on how to use customer segments to filter information to assign payments and billing, see the following:

- "Related Documents" in *BRM Email Manager*
- "Configuring Payment Fees" in *BRM Configuring and Collecting Payments*
- "Configuring Payment Incentives" in *BRM Configuring and Collecting Payments*

## 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 8–1 lists customer segments you can use to implement the following rules for early bill payers who rarely make late payments:

***Table 8–1 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.<br>**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. The customer segments are defined in the **/account** object PIN_FLD_CUSTOMER_SEGMENT_LIST field.

The customer segment rules apply to all accounts that belong to the customer segment.

To assign customer segment information to an account during account creation, see "Implementing Customer Segment Information".

# Defining Customer Segments in BRM

The following procedures explain how to define customer segments in your BRM database:

- Editing the pin_customer_segment.xml File
- Loading Customer Segments into BRM
- Validating the pin_customer_segment.xml File

To assign customer segment information to an account during account creation, see "Implementing Customer Segment Information".

## Editing the pin_customer_segment.xml File

You configure customer segments in the *BRM_Home*/**sys/data/config/pin_customer_segment.xml** file, where *BRM_Home* is the directory in which you installed the BRM software.

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 8–2:

***Table 8–2 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.<br><br>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.**<br><br>**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.<br><br>Maximum length is 1023 characters.<br><br>**Note:** This string is mapped to the PIN_FLD_DESCR field in the **/config/customer_segment** object. |

## Loading Customer Segments into BRM

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. See "load_pin_customer_segment".

**Important:** The **load_pin_customer_segment** utility needs a configuration file (**pin.conf**) in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

**Caution:** 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.

   See "Editing the pin_customer_segment.xml File" for more information.

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
   ```

   **Important:** 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). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

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*.

## Validating the pin_customer_segment.xml File

You run the "load_pin_customer_segment" utility with the **-t** parameter to validate the XML schema in your file.

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.

For more information, see "About Validating XML Configuration Files" in *BRM System Administrator's Guide*.

## Implementing Customer Segment Information

Customer Center and Self-Care Manager do not support creating customer segments. You can create your own application or use a third-party client application to assign the 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 outside of Customer Center are assigned a customer segment value of **0**.

# 9

# Managing Customer Billing Information

This chapter describes how to manage customer billing information, such as billing dates, credit limits, and payment methods.

For more information on Oracle Communications Billing and Revenue Management (BRM) billing, see "About Billing" in *BRM Configuring and Running Billing*.

## Changing Invoice Information

In addition to the main account contact, you can use a separate name and address for the person who receives the invoice. For example, if a company has several customer accounts, but only one person handles the invoice, you can send all invoices to one person.

You can change the following invoice information:

- The mailing address for the invoice. If the invoice recipient is the same as the account holder, this information should be the same as the customer contact information.
- How to deliver the invoice (postal mail or email). If you choose email, you enter the email address.

If you want the account holder's information to reflect the new billing information, update the contact information for the account after you have updated the invoice information.

See "Customizing Customer Payment Information" for more information on changing a customer's invoice information.

## Changing Credit Card Information

When you change credit card information, BRM validates the credit card. If credit card tokenization is enabled, BRM requests Paymentech for tokens and stores the tokens instead of the actual credit card numbers in the BRM database. See "About BRM-Initiated Payment Processing" in *BRM Configuring and Collecting Payments*.

---

**Important:** For security reasons, the Visa CVV2 numbers and American Express CID numbers are not stored in the BRM database; however, you can configure BRM to send them to the credit card processor when a credit card is added to an account or changed in an account if they are required for authorization. See "Requiring Additional Protection against Credit Card Fraud" in *BRM Configuring and Collecting Payments*.

---

The billing information for accounts is typically listed as the primary account contact. If you want the account holder's information to reflect the new billing information, update the contact information for the account after you have updated the credit card information.

Use Customer Center to change credit card information. See the discussion on changing credit card information in the Customer Center Help.

## Changing Direct Debit Information

The billing information for accounts is typically listed as the primary account contact. If you want the account holder's information to reflect the new billing information, update the contact information for the account after you have updated the debit card information.

Use Customer Center to change direct debit information. See the discussion on changing direct debit information in the Customer Center Help.

## Changing a Customer's Payment Method

Each account's bill unit includes a payment method, such as credit card or invoice.

To make an account's bill unit a nonpaying bill unit, add the account to an account group hierarchy as a child account. You can change an account's bill unit to a nonpaying bill unit only if the account was billed for the previous cycle.

When changing to the credit card payment method, BRM validates the customer's credit card. However, BRM does not charge the customer for the current balance until the end of the current billing cycle.

Use Customer Center to change payment methods. See the discussion on reviewing and modifying payment method information in the Customer Center Help.

### About Changing an Account's Bill Unit to the Nonpaying (Subordinate) Payment Method

If you change a child account bill unit's payment method to nonpaying (subordinate) 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.

See the discussion on moving an account between hierarchical groups in the Customer Center Help.

### About Changing a Closed Child Account's Bill Unit to the Nonpaying (Subordinate) Payment Method

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.

See the discussion on moving an account between hierarchical groups in the Customer Center Help.

### Changing the List of Payment Methods

Each account includes one or more payment methods, such as credit card or invoice. The list of available payment methods displayed in Customer Center is stored in the **/config/payment** object. Creating a new payment method requires programming. See "Customizing Payment Methods".

## Customizing Customer Payment Information

Customer payment information is stored in **/payinfo** objects. Use the PCM_OP_CUST_SET_PAYINFO opcode to create or modify a **/payinfo** object.

PCM_OP_CUST_CREATE_PAYINFO is called by the PCM_OP_CUST_UPDATE_CUSTOMER opcode when a customer's payment information is modified. For more information, see "Modifying an Account".

PCM_OP_CUST_SET_PAYINFO is a wrapper opcode that calls the following opcodes:

- PCM_OP_CUST_CREATE_PAYINFO, which creates the account **/payinfo** object.
- PCM_OP_CUST_MODIFY_PAYINFO, which modifies a **/payinfo** object.

To customize how **/payinfo** objects are created, use the following opcodes:

- PCM_OP_CUST_POL_PREP_PAYINFO. See "Customizing Payment Method Data Preparation" and "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.
- PCM_OP_CUST_POL_VALID_PAYINFO. See "Customizing Payment Method Validation" and "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.
- PCM_OP_CUST_FIND_PAYINFO. See "Finding Payment Info" in *BRM Configuring and Collecting Payments*.
- PCM_OP_PYMT_POL_SPEC_VALIDATE. See "Customizing the Account Used for Credit Card Validation".

To delete a **/payinfo** object, use the PCM_OP_CUST_DELETE_PAYINFO opcode. This opcode is given the **/payinfo** object Portal object ID (POID) of the object to delete.

> **Note:** You cannot delete a **/payinfo** object that is currently associated with a **/billinfo** object; you must first delete the **/billinfo** object.

If you have customized objects related to payment for an account, then, before you delete the **/payinfo** object for the account, you need to take steps to ensure that the deletion will not affect any customized information associated with that payment setup. To do so, use the PCM_OP_CUST_POL_PRE_DELETE_PAYINFO policy opcode.

The PCM_OP_CUST_POL_PRE_DELETE_PAYINFO policy opcode is called by the PCM_OP_CUST_DELETE_PAYINFO opcode before the latter opcode deletes the **/payinfo** object.

The PCM_OP_CUST_POL_PRE_DELETE_PAYINFO policy opcode requires the POID of the **/payinfo** object in the PIN_FLD_POID field of its input flist. It returns the POID of the **/payinfo** object in the PIN_FLD_POID field to the calling PCM_OP_CUST_DELETE_PAYINFO opcode.

For more information on the PCM_OP_CUST_POL_PRE_DELETE_PAYINFO policy opcode, see *BRM Developer's Reference*.

## Customizing Payment Method Data Preparation

The PCM_OP_CUST_POL_PREP_PAYINFO policy opcode processes inherited fields and prepares a **/payinfo** object. This policy opcode checks the payment method and creates the correct object based on that information.

- If the payment method is **invoice**, the **/payinfo/invoice** object is created.
- If the payment method is **cc** (credit card), the **/payinfo/cc** object is created and the information is prepared for online registration.
- If the payment method is **dd** (US and Canadian direct debit), the **/payinfo/dd** object is created and the information is prepared for online registration.
- If the payment method is **SEPA**, the **/payinfo/sepa** object is created and the information is prepared for SEPA payments.

If the payment method is **cc** or **dd**, the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode checks the **/config/ach** object to find the payment processor vendor to associate with the payment method. To use a vendor other than the default, you must customize this policy opcode.

### Specifying the Payment Processor Vendor

The PCM_OP_CUST_POL_PREP_PAYINFO policy opcode selects the first payment processor vendor listed in the **/config/ach** object to process the BRM-initiated payment. If you configured multiple payment processor vendors, you can specify the vendor to use by passing the vendor's ID in the PIN_FLD_ACH field in the opcode input flist. The vendor's ID corresponds to the ID of the PIN_FLD_ACH_INFO array, in the **/config/ach** object, that contains the information for that vendor.

## Customizing Payment Method Validation

The PCM_OP_CUST_POL_VALID_PAYINFO policy opcode validates inherited fields for a **/payinfo** object, which may include a **/payinfo/cc** object for credit cards, or a **/payinfo/dd** object for direct debit, or a **/payinfo/sepa** object for SEPA Direct Debit or SEPA Credit Transfer transactions.

For credit cards, this policy opcode checks the credit card type, number, and expiration date during registration.

For SEPA Direct Debit and Credit Transfer, this policy opcode checks that the International Bank Account Number (IBAN) and the Business Identifier Code (BIC) formats comply with the standards in the SEPA Rulebook and also validates that the primary currency of the account is euro.

> **Note:** Visa and American Express require a CVV2 or CID number for credit card fraud prevention. For security reasons, these numbers are not stored in the BRM database. For more information on how BRM handles these numbers, see "CVV2/CID Fraud Prevention Functionality".

If the information is valid, the standard checksum operation is performed.

### Default /payinfo Validation

This operation validates inherited fields for a **/payinfo** object according to the criteria contained in the **/config/fld_validate** object.

Mandatory fields for validation:

- PIN_FLD_CITY
- PIN_FLD_STATE
- PIN_FLD_ZIP
- PIN_FLD_COUNTRY
- PIN_FLD_CARD_TYPE

Checks performed for credit card-specific billing information:

- Is the debit number acceptable? (checksum)
- Is the debit number an acceptable type?
- Has the expiration date passed?

Checks performed for Paymentech direct debit transactions:

- Is the bank number acceptable? (9 digits)
- Is the debit number acceptable? (11 digits)
- What is the account type? (checking, corporate checking, savings)

### CVV2/CID Fraud Prevention Functionality

As a fraud prevention feature, Visa and American Express use additional three- and four-digit security codes attached to standard credit card numbers.

- Visa uses a three-digit CVV2 number in the signature section on the back of the card.
- American Express uses a four-digit CID number.

For security reasons, the PCM_OP_CUST_CREATE_PAYINFO and the PCM_OP_CUST_MODIFY_PAYINFO opcodes omit the PIN_FLD_SECURITY_ID field from the input flist of the PCM_OP_CREATE_OBJ opcode when the **/payinfo** object is created or changed. The result is that the CVV2/CID information is stored in the database with a NULL value. Likewise, the PCM_OP_PYMT_CHARGE opcode omits this value when a credit card is charged or validated; therefore, the **/event/billing/charge/cc** and **/event/billing/validate/cc** objects also store a NULL value for the PIN_FLD_SECURITY_ID value.

If the Connection Manager (CM) **pin.conf** file's **cvv2_required** flag is set to **1** (required), these opcodes send the PIN_FLD_SECURITY_ID value directly to the credit card processor when customers add or change credit cards in their accounts. If the CVV2 information is not provided in the input flist, the PIN_FLD_RESULT value is set to PIN_ERR_MISSING_ARG, with the description "Missing argument".

For more information on how to make these values required or optional, see "Requiring Additional Protection against Credit Card Fraud" in *BRM Configuring and Collecting Payments*.

### Verifying the Maximum Number of CVV2 Digits

By default, the PCM_OP_CUST_POL_VALID_PAYINFO policy opcode verifies that a credit card's CVV2 passed in the PIN_FLD_SECURITY_ID input flist field does not

exceed three digits. You can modify the maximum allowed number of CVV2 digits by customizing the policy opcode to:

- Validate that the number of digits passed in the PIN_FLD_SECURITY_ID input flist field of the PIN_FLD_CC_INFO array does not exceed a specified value.
- If the validation succeeds, pass the PIN_FLD_SECURITY_ID field to the calling opcode.
- If the validation fails, return the appropriate error in the PIN_FLD_RESULT output flist field.

### Disabling the Credit Card Checksum

To disable the credit card checksum:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**, where *BRM_Home* is the directory in which BRM is installed).
2. Change the flag in the **cc_checksum** entry from **1** (enable) to **0** (disable):

   ```
   - fm_cust_pol cc_checksum 0
   ```

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.

### Customizing the Banking Information for US and Canadian Direct Debit

The bank number (**/bank_no**) and debit number (**/debit_num**) fields can be customized to allow for bank and debit numbers that do not meet the US standard (5 digits) by using the Field Validation Editor in Configuration Center.

### Customizing the Account Used for Credit Card Validation

When validating a credit card at registration, BRM needs an account to validate the card with. By default, this is the root account. You cannot store this information with each account because the credit card validation is done before the account is created.

You can use the PCM_OP_PYMT_POL_SPEC_VALIDATE policy opcode to change the account used for credit card validation.

The default account is:

```
0.0.0.1 /account 1
```

You can change the **1** to some other account number as long as it has a bill type that is not affected by billing operations or events.

The PCM_OP_PYMT_POL_SPEC_VALIDATE policy opcode reads the following CM **pin.conf** file entries:

- The **cc_validate** entry, which specifies whether to validate credit cards. See "Validating Credit Cards at Registration".
- The **cc_revalidate** entry, which specifies the amount of time before revalidation is required. See "Revalidating Credit Cards".

In addition, the PCM_OP_PYMT_POL_SPEC_VALIDATE policy opcode reads the **/config/ach** object to find the merchant value.

The only types of action supported are **prep customer** and **commit customer**. If one of these is not passed in, PIN_ERR_BAD_VALUE is returned. If the action passed in is NULL, a blank flist is returned. If an error occurs, a null flist is returned.

If the PIN_FLD_PAY_TYPE value is NULL, only the POID from the **pin.conf** file is returned. If the PIN_FLD_PAY_TYPE value is not PIN_PAY_TYPE_CC or PIN_PAY_TYPE_DD, the result is set to PIN_BOOLEAN_FALSE because there is nothing to validate. If the value is PIN_PAY_TYPE_CC or PIN_PAY_TYPE_DD, the result is set to PIN_BOOLEAN_TRUE.

## Customizing Payment Methods

This section describes the information necessary to add custom payment methods to your BRM system.

### Understanding Payment Methods

The PIN_FLD_PAY_TYPE field in the **/billinfo** object defines the payment method for an account's bill. All defined payment methods are stored in the **/config/payment** object.

When you add a new payment method to your BRM system, you must update the PIN_FLD_PAY_TYPES array in the **/config/payment** object. The PIN_FLD_PAY_TYPES array index corresponds to an entry in the **pin_pymt.h** file, where payment methods are defined.

---

**Note:** If you create accounts that have custom payment methods, you must modify the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode to validate them. For example, add code for your custom payment method everywhere the opcode checks the various payment methods.

---

Use the information in the following tables to edit the predefined payment methods and element IDs in the **/config/payment** object:

- Payment Methods and Element IDs
- Opcodes Using PIN_FLD_PAY_TYPE
- Fields in PIN_FLD_PAY_TYPE

See **pin_pymt.h** in the *BRM_Home*/**include** directory for a complete list of supported payment methods.

Table 9–1 list the payment methods and element IDs.

***Table 9–1 Payment Methods and Element IDs***

| Payment Method | Element ID |
|---|---|
| PIN_PAY_TYPE_UNDEFINED<br>Used during account creation. | 0 |
| PIN_PAY_TYPE_PREPAID<br>Used to keep negative balances. | 10000 |
| PIN_PAY_TYPE_INVOICE<br>Used for monthly invoices. | 10001 |

*Table 9–1 (Cont.) Payment Methods and Element IDs*

| Payment Method | Element ID |
|---|---|
| PIN_PAY_TYPE_DEBIT<br>Used for checking account debit. | 10002 |
| PIN_PAY_TYPE_CC<br>Used for credit cards. | 10003 |
| PIN_PAY_TYPE_DD<br>Used for US/Canadian direct debits. | 10005 |
| PIN_PAY_TYPE_SMARTC<br>Used for smartcards. | 10006 |
| PIN_PAY_TYPE_SUBORD<br>Used to roll up balances to the parent account. | 10007 |
| PIN_PAY_TYPE_BETA<br>For use by beta testers only. Billing utilities ignore this. | 10008 |
| PIN_PAY_TYPE_INTERNAL<br>Used for internal employees. Used the same way as guest accounts. | 10009 |
| PIN_PAY_TYPE_GUEST<br>Used for guest accounts. It is not charged, but credit limits apply. | 10010 |
| PIN_PAY_TYPE_CASH<br>Used for cash. | 10011 |
| PIN_PAY_TYPE_CHECK<br>Used for personal bank checks. | 10012 |
| PIN_PAY_TYPE_WTRANSFER<br>Used for wire transfers. | 10013 |
| PIN_PAY_TYPE_PAYORDER<br>Used for inter-bank payment orders. | 10014 |
| PIN_PAY_TYPE_POSTALORDER<br>Used for postal payment orders. | 10015 |
| PIN_PAY_TYPE_VOUCHER<br>Used for payment vouchers. | 10016 |
| PIN_PAY_TYPE_FAILED<br>Used for unconfirmed payments that failed. | 10017 |
| PIN_PAY_TYPE_SEPA<br>Used for SEPA payments. | 10018 |

**Note:** To avoid conflicts with payment IDs reserved by BRM, custom payment methods should be defined with an element ID greater than 10099.

**Important:** If a customer makes a payment with a payment method that has not been saved with an account, you must add the new payment method, when first used, to the input flist of the following opcodes. For example, this is required if an invoice customer makes a one-time payment by using a new credit card. (This is not required if the payment method was defined when the account was created. If the new payment method is SEPA, you must add the payment method to the account before it can be used.)

Table 9–2 lists the opcodes that use the PIN_FLD_PAY_TYPE field.

***Table 9–2 Opcodes Using PIN_FLD_PAY_TYPE***

| Opcode Name |
|---|
| PCM_OP_PYMT_CHARGE |
| PCM_OP_PYMT_CHARGE_CC |
| PCM_OP_PYMT_COLLECT |
| PCM_OP_BILL_GROUP_DELETE_MEMBER |
| PCM_OP_BILL_MAKE_BILL |
| PCM_OP_BILL_MAKE_BILL_NEW |
| PCM_OP_BILL_MAKE_BILL_ON_DEMAND |
| PCM_OP_BILL_MAKE_BILL_RCV_PAYMENT |
| PCM_OP_PYMT_RECOVER |
| PCM_OP_BILL_REVERSE |
| PCM_OP_BILL_REVERSE_PAYMENT |
| PCM_OP_PYMT_VALIDATE |

Table 9–3 lists the fields in PIN_FLD_PAY_TYPES array.

***Table 9–3 Fields in PIN_FLD_PAY_TYPE***

| Field Name | Description |
|---|---|
| PIN_FLD_PAYINFO_TYPE | A string that indicates the type of **/payinfo** object to create. |
| PIN_FLD_PAYMENT_EVENT_TYPE | A string that indicates the type of **/event** object to create when payment is received. |
| PIN_FLD_REFUND_EVENT_TYPE | A string that indicates the type of **/event** object to create when a refund is paid. |

*Table 9–3 (Cont.) Fields in PIN_FLD_PAY_TYPE*

| Field Name | Description |
|---|---|
| PIN_FLD_OPCODES | An array that contains the following information about an opcode to be associated with this pay type:<br>■ PIN_FLD_NAME<br>The name of the opcode to be associated with this pay type.<br>■ PIN_FLD_OPCODE<br>The number of the opcode to be associated with this pay type.<br>■ PIN_FLD_EVENT_TYPE<br>A string that indicates the type of **/event** object to create.<br>■ PIN_FLD_FLAGS<br>Flags passed to opcodes when executed. |

Table 9–4 lists the PIN_FLD_OPCODES array indexes-to-opcode mapping.

*Table 9–4 Index to Code Mapping for PIN_FLD_OPCODES*

| Value/Element ID | Opcode Type |
|---|---|
| 0 | PIN_PAY_VALIDATE |
| 1 | PIN_PAY_CHARGE |
| 2 | PIN_PAY_RECOVER |
| 3 | PIN_BILL_REVERSAL |

## Updating the /config/payment Object

To add a custom payment method to your BRM system, edit the **/config/payment** object.

1. Using a text editor, edit the file to add an element to the PIN_FLD_PAY_TYPES array for each new payment method.
   - For a predefined payment method, use the appropriate value as the index for that element of the PIN_FLD_PAY_TYPES array. See " Payment Methods and Element IDs".
   - For a custom payment method, define a payment method in a header file and use that value as the index of the new element in the PIN_FLD_PAY_TYPES array.

     **Important:** To avoid conflicts with payment IDs used by BRM, define custom payment methods with an element ID of 10100 or higher.

2. Save and close the file.
3. Use the **testnap** utility to load the **/config/payment** object into the database. See "Testing Your Applications and Custom Modules" in *BRM Developer's Guide*.

### Creating a New /config/payment Object

You can configure your own **/config/payment** object and load it into the database using the **testnap** utility. See "Creating Objects" in *BRM Developer's Guide*.

After you create the **/config/payment** object, update the **config_payment** entry in the CM **pin.conf** file with the POID of the new object:

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

   ```
   - fm_pymt config_payment database_number /config/payment 200
   ```

   where *database_number* is the database number of the BRM database. By default, this number is **0.0.0.1**. Replace **200**, the default POID of the **/config/payment** object, with the POID of the new object.
3. Save and close the file.
4. Stop and restart the CM.

### Viewing Payment Information for a Custom Payment Method

To view payment information in Customer Center for custom payment methods, run the **pin_collect** utility for that payment method. The **pin_bill_day** utility calls **pin_collect** for credit card payment methods and **pin_inv_accts** for invoice payment methods. You can make a call to **pin_collect** in **pin_bill_day** for each new custom payment method.

For more information, see "Running Daily Billing" (in *BRM Configuring and Running Billing*) and "About Collecting BRM-Initiated Payments" (in *BRM Configuring and Collecting Payments*).

### Using the Undefined Payment Method

You can set a bill unit's payment method to **Undefined** to enable users to access services without being charged. When you set the payment method for a bill unit to **Undefined**, BRM generates rating events and applies credit limits in the same way it does for a normal payment method; however, a payment is not requested from the account.

You can set the payment method to **Undefined** by either:

- Using Customer Center.
- Calling the PCM_OP_CUST_SET_BILLINFO opcode in an application and setting the value of PIN_FLD_PAY_TYPE to PIN_BILL_TYPE_UNDEFINED (0).

**Important:** By default, credit limits are set to **0**. Because an account with a payment method of **Undefined** never pays a bill, you need to set the credit limit to **Unlimited**.

## Changing a Customer's Billing Type

Accounts that pay by credit card should always use balance forward accounting. Accounts that use other payment methods can use either balance forward accounting or open item accounting.

For more information, see "About Accounting Types" in *BRM Configuring and Running Billing*.

# Changing a Customer's 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*.

The default start date for an accounting cycle is the account's creation date. To change that date, see "Setting the Default Accounting Day of Month (DOM)" in *BRM Configuring and Running Billing*.

Use Customer Center to change the billing day of month. See the Customer Center Help.

If you use bill cycle management, see "Changing a Bill Unit's Billing DOM" for more information on changing a customer's billing day of month. For an overview of bill cycle management, see "About Managing Billing Cycles".

## Prorating Fees for a Partial Cycle

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

# Changing a Customer's Billing Cycle Length

Use Customer Center to change a customer's billing cycle length. See "About Accounting and Billing Cycles" in *BRM Configuring and Running Billing* and the discussion on reviewing or changing the billing frequency or billing date in the Customer Center Help.

**Note:**

- Because billing cycles are whole multiples of accounting cycles, the only cycle fee proration that might occur is for multi-month cycle rates.
- A child account with a subordinate bill unit must use the same currency, billing frequency, billing day of the month, and accounting type as its parent account.

# Changing a Customer's Bill Due Date

Use the bill run management feature to change a customer's bill due date. See "About Managing Bill Due Dates" in *BRM Configuring and Running Billing*.

# Changing a Customer's Credit Limit

A *credit limit* is the maximum amount of a resource, such as currency or hours, that can accumulate in an account balance group before the customer is prevented from using the service. By default, a customer cannot be authorized for service usage if a credit limit has been reached.

**Note:** For nonpaying bill units, BRM uses the credit limit of the parent (paying) accounts receivable (A/R) bill unit to determine whether the nonpaying bill unit's credit limit is reached.

You define credit limits in plans. For example, a plan might include a credit limit of $200. You change an individual customer's credit limit by using Customer Center.

You can also define credit thresholds to alert customers when the credit limit is about to be reached. You use event notification to alert customers automatically. See "About Applying Credit Limits to Resources" in *BRM Setting Up Pricing and Rating*.

You can set credit limits for currency resources and noncurrency resources.

**Note:** If the account uses a secondary currency, you need to set the credit limit in the primary currency. See "Managing Currencies in Customer Accounts".

Use Pricing Center to set credit limits for plans. See the Pricing Center Help.

Use Customer Center to set or change credit limits. See the Customer Center Help.

## Handling Credit Limit Conflicts

There might be occasions when customers purchase plans with credit limits that conflict with the credit limit set on their account or another service. There are two ways to handle credit limit conflicts:

- When you create your price plans, if two services have different credit limits, create new balance groups for the services. This allows credit limits for each service to be set and tracked independently. See "Tracking Resources by Service" in *BRM Setting Up Pricing and Rating*.
- Specify which credit limit takes precedence when a new credit limit conflicts with an existing credit limit. You do this by setting the **credit_limit_conflict** entry in the CM configuration file.

To specify credit limit precedence:

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

   ```
   -fm_bill credit_limit_conflict value
   ```

   where *value* is one of the following:

   - **replace**, to use the new credit limit.
   - **ignore**, to ignore the new credit limit and keep the existing credit limit.
   - **add**, to add the new credit limit to the existing credit limit and create a new limit.
   - **minimum**, to use the credit limit that specifies the smaller amount.
   - **maximum,** to use the credit limit that specifies the greater amount.

   If this entry is not present, the new credit limit is used by default.
3. Stop and restart the CM.

## Ignoring Credit Limits during the Rating Process

BRM automatically checks whether an account balance has exceeded a credit limit when processing authorization requests or rating events. When an account balance exceeds a credit limit, BRM:

- **Denies all authorization requests**. This prevents an account from logging in or using services until the account balance falls below the credit limit.
- **Stops rating events associated with the account**. BRM rates events up to the credit limit amount and then returns any remaining usage as unrated.

To prevent revenue leakage, you can configure BRM to rate all events as defined in the rate plan, regardless of whether the account balance exceeded the limit.

You configure whether BRM ignores credit limit settings during the rating process in three different ways as shown in Table 9–5.

***Table 9–5 Ways to Override Credit Limit Settings***

| Override | Event Type | Description |
|---|---|---|
| Product-level override | Actual event | Specifies to ignore credit limit settings when rating the specified product. When a resource has exceeded a credit limit, BRM uses the credit limit override rate to rate the specified product.<br>This product-level setting takes precedence over the system-wide default setting.<br>See the discussion on defining rates in the Pricing Center Help.<br>**Note:** This setting does not affect prepaid AAA events, which are always checked against the credit limit. |
| System-wide default | Actual event | Specifies to ignore the credit limit setting during the rating process.<br>See "Configuring the System-Wide Override Credit Limit".<br>**Note:** This setting does not affect prepaid AAA events, which are always checked against the credit limit. |
| Calc-only override | Calc-only event | Specifies to ignore credit limit settings during calc-only rating. You use this setting for calc-only events that do not require credit limit checks, such as for best pricing rating or prepaid top-ups.<br>You override credit limit checks during calc-only rating by passing the PIN_RATE_FLG_OVERRIDE_CREDIT_LIMIT flag in the PIN_FLD_FLAGS input flist field of an opcode. |

**Note:** Even when BRM is configured to enforce all credit limit settings, BRM may still ignore credit limits during real-time pipeline discount rating and charge sharing.

### Configuring the System-Wide Override Credit Limit

By default, BRM stops rating any events for accounts that have exceeded their credit limit. To prevent revenue leakage, you can configure BRM to ignore credit limits during the rating process and rate all events, regardless of whether a credit limit was breached.

You specify whether BRM enforces or ignores credit limit settings during the rating process by using the **OverrideCreditLimit** entry in the **/config/business_params** object.

- When this feature is enabled, the rating engine applies the full cycle-forward fee and full usage fee.
- When this feature is disabled, the rating engine applies part of the cycle-forward fee or usage fee until the credit limit is reached.

> **Note:** Tax amounts are applied even if the credit limit has been exceeded and the **OverrideCreditLimit** entry is disabled.

To set credit limit enforcement:

1. Go to the *BRM_Home*/**sys/data/config** directory and run the following command:

   ```
   pin_bus_params -r BusParamsRating bus_params_rating.xml
   ```

   This command creates an editable 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. Search the XML file for the following line:

   ```
   <BusParamsRating>
        <OverrideCreditLimit>disabled</OverrideCreditLimit>
   </BusParamsRating>
   ```

3. Set the **OverrideCreditLimit** entry to the appropriate value:
   - **enabled** specifies to *ignore* all account credit limit settings during the rating process. BRM continues to enforce credit limits during the authorization process.
   - **disabled** specifies to *enforce* credit limit settings. This is the default setting.
4. Save and close the file.
5. Go to the *BRM_Home*/**sys/data/config** directory and run the following command:

   ```
   pin_bus_params bus_params_rating.xml
   ```

   This command loads your changes into the **rating** instance of the **/config/business_params** object.

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

6. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
7. Stop and restart the CM.
8. (Multischema systems only) Run the **pin_multidb** script:

   ```
   pin_multidb -R CONFIG
   ```

   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

# Changing a Customer's Credit Threshold

A credit threshold is a resource balance value, such as 90 minutes or $150, that triggers an alert to the customer. This allows customers to know when their balances are approaching a credit limit and to adjust their spending habits accordingly.

BRM can perform credit threshold checks during both real-time rating and batch rating:

- Real-time rating: Credit threshold checks are performed for both currency and non-currency resources. To configure real-time rating to perform credit threshold checks, see "Configuring Event Notification for Threshold Checking".
- Batch rating: Credit threshold checks are performed for only non-currency resources, such as minutes. To configure batch rating to perform credit threshold checks, see "Configuring Event Notification for Threshold Checking" and "Setting Up the Batch Rating Process to Perform Threshold Checking".

> **Note:** Performing credit threshold checking during batch rating decreases pipeline processing performance. See "Enabling Threshold Checking in Pipeline Manager".

## About Setting Credit Threshold Values

Credit thresholds can be specified either as a percentage of the credit floor and credit limit range or as a fixed value.

For example, you can set a credit threshold to 90 minutes in two different ways:

- By setting the credit floor to 0, the credit threshold to 90%, and the credit limit to 100 minutes.
- By setting the credit threshold to 90 minutes.

In the above example, if credit floor is set to 10 and the credit threshold is 90%,

- The threshold will be 100, for a credit limit of 110.
- The threshold will be 91, for a credit limit of 100.

You set or modify credit threshold values in your plans by using Pricing Center. After a customer purchases a plan, you can change the customer's individual credit threshold values by using Customer Center.

> **Note:** Customer Center can modify only percentage threshold values. To modify fixed threshold values, you must customize your client application to call the PCM_OP_BILL_SET_LIMIT_AND_CR opcode. See "Customizing Client Applications to Modify Fixed Thresholds".

## About Alerting Customers When Credit Thresholds Are Breached

BRM uses event notification to alert customers when their balances breach a credit threshold value. When a threshold is breached, BRM generates a notification event, which triggers a call to a specified opcode. The opcode can then send an alert to the client application, email the customer, or perform other custom actions. For more information, see "Using Event Notification" in *BRM Developer's Guide*.

To indicate that a resource balance breached a customer's credit threshold, BRM generates one of the following notification events:

- **/event/notification/threshold** when the balance crosses above the credit threshold value due to a new balance impact.
- **/event/notification/threshold_below** when the balance falls below the credit threshold value due to a customer payment or an adjustment.

These notification events trigger a call to the opcode specified in the **/config/notify** object. By default, the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode is called. This policy opcode processes the event data and passes information to Customer Center or your custom client application. You can customize the policy opcode to perform other actions, such as sending an email to the customer.

To alert customers when credit thresholds are breached, you must configure the event notification feature, the policy opcode, and your custom client application to process these events and alert customers. For more information, see "Configuring Event Notification for Threshold Checking".

## When Credit Thresholds Are Checked

BRM performs credit threshold checking at the following times:

- When balance impacts are applied during real-time rating. For more information, see "About Credit Limit and Threshold Checking during Real-Time Rating".
- When balance impacts are applied during batch rating. For more information, see "About Credit Limit and Threshold Checking during Batch Rating".

  **Note:** The utility that loads balance impacts into the BRM database is different from the utility that loads threshold breach notifications into BRM. Therefore, the customer's balance may not immediately reflect the balance on the threshold breach notification.

- When a credit profile is modified *and* balance monitoring is enabled. After a customer's credit threshold values are updated in the credit profile, BRM determines whether the customer's balance has breached any of the new threshold values.

## About Credit Limit and Threshold Checking during Real-Time Rating

During the real-time rating process, BRM checks for threshold breaches after calculating the charges generated by an event and applying the balances but before the customer's balances are updated in the database. BRM compares the customer's new balance against the customer's credit threshold values. If a balance crossed a credit threshold value, BRM generates a notification event.

If balance monitoring is enabled, BRM also checks for credit threshold breaches when a credit profile is modified. The PCM_OP_BILL_SET_LIMIT_AND_CR opcode compares the customer's current balance to the updated credit floor, credit limit, and credit threshold values that are stored in the customer's credit profile (**/config/credit_profile** object). If the balance has breached a credit threshold value, BRM generates a notification event.

To enable custom client applications to perform credit threshold checks during the real-time rating process, configure your client application to pass information in the input flist of the PCM_OP_BILL_SET_LIMIT_AND_CR opcode. For more information,

see "How BRM Handles Consumption Rules and Credit Limits".

## About Credit Limit and Threshold Checking during Batch Rating

During batch rating, Pipeline Manager determines whether a customer's *noncurrency* balance has breached a credit threshold. Pipeline Manager modules compare a customer's balance against non-currency credit threshold values and then compile a list of events that crossed thresholds. A utility loads the list from Pipeline Manager into BRM.

**Important:** Pipeline rating calculates balance impacts and credit threshold breaches immediately. However, the utility that loads the balance impacts into the BRM database is different from the utility that loads credit threshold breaches into BRM. This means that a notification for a credit threshold breach might be sent before the customer's balance is updated in the BRM database. Therefore, the customer balance shown on the threshold breach notification might temporarily differ from the balance in the customer's account.

The following pipeline modules are responsible for checking credit thresholds during the batch rating process:

- DAT_PortalConfig retrieves credit profile information from the BRM database and stores it in its internal memory.
- DAT_BalanceBatch retrieves the customer's current credit profile from the DAT_PortalConfig module, compares the customer's updated balance to the credit profile, and notifies FCT_ApplyBalance if a credit threshold or credit limit has been breached.
- FCT_ApplyBalance updates the customer's current balance in the DAT_BalanceBatch internal memory and, if a credit threshold has been breached, writes the event detail record (EDR) into an internal list.

  To help you identify the exact event that caused the threshold breach, FCT_ApplyBalance adds the following event attributes to each threshold breach notification:

  - Event type (DETAIL.EVENTTYPE)
  - Call originator (DETAIL.A_NUMBER)
  - Call destination (DETAIL.B_NUMBER)

  At the end of the transaction, FCT_ApplyBalance writes all of the EDRs from the internal list to an XML output file. See "About the Format of the XML Output File Name".

After the XML output file is generated, Batch Controller uses the **load_notification_event** utility to load the notification events into BRM. See "About Loading Notifications from Pipeline Manager to BRM".

### About the Format of the XML Output File Name

Pipeline Manager generates output XML files using the following file name format:

*OutputPrefix_PipelineName_StreamName_TransactionID_SequenceNumber*

where:

- *OutputPrefix* is the prefix name specified in the **OutputPrefix** registry entry.
- *PipelineName* is the name of the individual pipeline, such as ALL_RATE.
- *StreamName* is based on the **unitsPerTransaction** parameter maintained at the input level of the pipeline. A value of **1** means that one input file generates one output file. A value greater than **1** means that multiple input files are merged into one output file. *StreamName* is replaced with the name of the last file merged into the output file. For example, if **unitsPerTransaction** was **3** and the files merged into the output file were **file01**, **file02**, and **file03**, *StreamName* would be replaced with **file03**.
- *TransactionID* is the system-generated transaction ID number.
- *SequenceNumber* signifies the order in which FCT_ApplyBalance created the XML output file within a transaction. For example, the first XML output file has a sequence number of 1, the second output file has a sequence number of 2, and so on.

For example:

```
balancenotification_ALL_RATE_file01_776_1
```

### About Loading Notifications from Pipeline Manager to BRM

Notifications from Pipeline Manager are loaded into the BRM database by the **load_notification_event** utility. This utility is launched automatically by Batch Controller whenever Pipeline Manager writes an output XML file to a specified directory.

When Pipeline Manager writes an output XML file to the specified directory, notifications are loaded as follows:

1. Batch Controller launches the SampleLoadHandler batch handler.
2. SampleLoadHandler moves the output XML file to a processing directory and runs the **load_notification_event** utility.
3. **load_notification_event** converts the notification events in the output XML file into flist format and loads them into BRM.
4. The BRM event notification system sends an alert to the customer. See "About Alerting Customers When Credit Thresholds Are Breached".
5. SampleLoadHandler determines whether the utility successfully loaded the notification events into BRM:
   - If the utility was successful, SampleLoadHandler moves the output XML file from the processing directory to the archive directory.
   - If the utility failed, SampleLoadHandler moves the output XML file from the processing directory to the reject directory.

You must configure Batch Controller, the batch handler, and the **load_notification_event** utility to process the output XML files. See "Configuring Batch Controller to Run load_notification_event".

## Customizing Client Applications to Modify Fixed Thresholds

To modify any fixed threshold values in a customer's credit profile, you must customize your client application to call the PCM_OP_BILL_SET_LIMIT_AND_CR opcode.

You pass the fixed threshold values in the input flist's PIN_FLD_THRESHOLDS array. Each fixed value has a separate PIN_FLD_THRESHOLD flist entry in the array.

When the opcode receives fixed threshold values in its input flist, PCM_OP_BILL_SET_LIMIT_AND_CR updates the amount, refreshes the cache of credit profiles, and generates a **ModifyBalanceGroup** business event for Pipeline Manager so DAT_BalanceBatch updates its cache as well.

## Configuring Event Notification for Threshold Checking

When a balance breaches a credit threshold value, BRM generates either an **/event/notification/threshold** or an **/event/notification/threshold_below** notification event. By default, when these events occur, the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode is called. You can either:

- Customize the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode to process threshold breaches. You can add custom code to this policy opcode so that it alerts the CRM client application or sends an email to the customer. For more information, see "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.
- Configure event notification to call a different opcode by editing the *BRM_Home***/sys/data/config/pin_notify** file and running the **load_pin_notify** utility. See "load_pin_notify". For more information, see "Implementing Event Notification" in *BRM Developer's Guide*.

## Setting Up the Batch Rating Process to Perform Threshold Checking

To set up your system to perform threshold checking during the batch rating process:

- Enable threshold checking. See "Enabling Threshold Checking in Pipeline Manager".
- Configure Batch Controller to run the **load_notification_event** utility. See "Configuring Batch Controller to Run load_notification_event".
- Configure Pipeline Manager to check thresholds. See "Configuring Pipeline Manager to Perform Threshold Checking".

### Enabling Threshold Checking in Pipeline Manager

By default, threshold checking is disabled for batch rating because it decreases pipeline rating performance.

You can enable threshold checking in batch rating by modifying a field in the **multi_bal** instance of the **/config/business_params** object.

To enable credit threshold checking during batch rating:

1. Go to the *BRM_Home***/sys/data/config** directory and run the following command:

   ```
   pin_bus_params -r BusParamsMultiBal bus_params_multi_bal.xml
   ```

   This command creates an editable XML file named **bus_params_multi_bal.xml.out** in your working directory. If you do not want this file in your working directory, specify the path as part of the file name.

2. Search the XML file for the following line:

   ```
   <CreditThresholdChecking>disabled</CreditThresholdChecking>
   ```

3. Set the **CreditThresholdChecking** entry to the appropriate value:

   - **enabledOffline** specifies to check credit thresholds during the batch rating process.

- **disabled** specifies to skip credit threshold checking during the batch rating process and thus increase pipeline processing performance. This is the default setting.

4. Go to the *BRM_Home***/sys/data/config** directory and run the following command:

   ```
   pin_bus_params bus_params_multi_bal.xml
   ```

   This command loads your changes into the **multi_bal** instance of the **/config/business_params** object.

   **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 **multi_bal** configuration.

5. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.
6. Stop and restart the CM.
7. (Multischema systems only) Run the **pin_multidb** script:

   ```
   pin_multidb -R CONFIG
   ```

   For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

### Configuring Batch Controller to Run load_notification_event

Batch Controller is a **cronjob** application that monitors a processing directory and starts batch handlers. When configured for threshold checking, Batch Controller waits for output XML files to appear in the processing directory and then launches the SampleLoadHandler handler. The batch handler moves the output XML files to a separate directory and then calls the **load_notification_event** utility to load the notifications into the BRM database. For more information on Batch Controller, see "Controlling Batch Operations" in *BRM System Administrator's Guide*.

You configure Batch Controller by using the following configuration files:

- The **batch_controller.properties** file connects the Batch Controller to the CM, specifies run-time parameters, identifies the batch handlers to run, and specifies when and how to run the batch handlers.
- The **load_notification_event pin.conf** file connects the **load_notification_event** utility to the CM.
- The **SampleLoadHandler_config.values** file specifies where the batch handler retrieves and deposits the XML files and which utility to run.

To configure Batch Controller for threshold checking:

1. Open the *BRM_Home***/apps/batch_controller/batch_controller.properties** file in a text editor.
2. Connect Batch Controller to the CM, if you have not already done so. See "General Batch Controller Parameters" in *BRM System Administrator's Guide*.
3. Make sure the entries for the **load_notification_event** batch handler (*BRM_Home***/apps/load_notification_event/SampleLoadUtilityHandler.pl**) are correct. For example:

   ```
   SampleHandler.name = SampleHandler
   ```

```
SampleHandler.max.at.lowload.time = 1
SampleHandler.start.string = $PIN_HOME/apps/load_notification_
event/SampleLoadUtilityHandler.pl
```

For more information, see "Handler Identification" in *BRM System Administrator's Guide*.

4. Specify the handler's polling directory and the file pattern to look for. For example:

```
event1.name = eventlabel1
event1.handlers = SampleHandler
event1.at = 115700
event1.file.location = $HOME/opt/ifw/data/out/notifybalancebreach/
event1.file.pattern = *.xml
```

For more information, see "Occurrence-Driven Execution" in *BRM System Administrator's Guide*.

5. Save and close the file.
6. Connect the **load_notification_event** utility to the BRM database by editing the *BRM_Home***/apps/load_notification_event/pin.conf** file. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.
7. Open the *BRM_Home***/apps/load_notification_event/SampleLoadHandler_config.values** file in a text editor.
8. Specify the directory from which to retrieve the XML files in the $STAGING entry. For example:

```
$STAGING = "$HOME/opt/ifw/data/out/notifybalancebreach/"
```

9. Specify the directory in which the handler archives the output XML files that loaded successfully into BRM in the $ARCHIVE entry. For example:

```
$ARCHIVE = "$HANDLER_DIR/archive"
```

10. Specify the directory in which the handler stores the output XML files that failed to load into BRM in the $REJECT entry. For example:

```
$REJECT = "$HANDLER_DIR/reject"
```

11. Save and close the file.
12. Start Batch Controller.

### Configuring Pipeline Manager to Perform Threshold Checking

To configure threshold checking, perform the following:

1. Configure the DAT_BalanceBatch module. See "DAT_BalanceBatch" in *BRM System Administrator's Guide*.
2. Configure the DAT_PortalConfig module. See "DAT_PortalConfig" in *BRM System Administrator's Guide*.
3. Configure the FCT_ApplyBalance module. See "FCT_ApplyBalance" in *BRM System Administrator's Guide*.

   When you configure the FCT_ApplyBalance module, you specify the following:

   - The maximum number of notifications in the output file. See "Configuring the Maximum Number of Notifications".

- The output file prefix and directory location. See "Configuring the Output XML File".

#### Configuring the Maximum Number of Notifications

You specify the maximum number of notification events that FCT_ApplyBalance writes into the XML output file by using the **NumberOfNotificationLimit** registry entry. When the file breaches the maximum number, FCT_ApplyBalance closes the file and begins writing notification events into a new XML output file.

> **Note:** You can identify the order in which XML output files were created by the sequence number used in the file name. See "About the Format of the XML Output File Name".

#### Configuring the Output XML File

After FCT_ApplyBalance identifies that an EDR caused a balance to breach a credit threshold, it writes the EDR to an output XML file. You can specify the prefix for the output file name by using the **OutputPrefix** registry entry and the directory in which to save the file by using the **OutputDirectory** registry entry.

## Customizing Credit Limits and Resource Consumption Rules

*Resource consumption rules* specify the order in which resource sub-balances are consumed. For example, when a customer is granted free minutes with different validity periods, you can specify which minutes to use first, based on the validity start time and end time.

A *credit limit* is the maximum amount of a resource, such as currency or hours, that can accumulate in an account balance group before the customer is prevented from using the service. By default, a customer cannot be authorized for service usage if a credit limit has been reached.

You use the following opcodes to set credit limits and consumption rules in an account:

- To set both the credit limit and the resource consumption rules, use the PCM_OP_BILL_SET_LIMIT_AND_CR opcode. See "How BRM Handles Consumption Rules and Credit Limits".
- To customize setting the credit limit, use the PCM_OP_CUST_POL_PREP_LIMIT policy opcode and the PCM_OP_CUST_POL_VALID_LIMIT policy opcode. See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

### How BRM Handles Consumption Rules and Credit Limits

For more information on how BRM uses consumption rules, see "Specifying the Order in Which Resource Sub-Balances Are Consumed" in *BRM Setting Up Pricing and Rating*.

Use PCM_OP_BILL_SET_LIMIT_AND_CR to set a credit limit and resource consumption rule in an account.

The opcode sets the consumption rule and credit limit in a balance group for both currency and non-currency resources. This opcode must be called separately for each resource to be set. PCM_OP_BILL_SET_LIMIT_AND_CR creates a new **/event/billing/limit** event each time a credit limit is changed and a new **/event/billing/consumption_rule** event each time a consumption rule is changed.

By default, PCM_OP_BILL_SET_LIMIT_AND_CR sets or changes the credit limit and consumption rule in the account-level **/balance_group** object. To set a credit limit and consumption rule for any of the other billing entities associated with the object, specify them with the optional PIN_FLD_BAL_GRP_OBJ input flist field.

The credit limit is passed in the PIN_FLD_LIMIT array of the PCM_OP_BILL_SET_LIMIT_AND_CR input flist, which contains the resource ID of the sub-balance to change or create. The credit limit is set in the PIN_FLD_BALANCES array of the **/balance_group** object of the various billing entities. The POID of this object is passed in the PIN_FLD_BAL_GRP_OBJ field of the input flist.

The consumption rule is passed in the PIN_FLD_RULES array of the PCM_OP_BILL_SET_LIMIT_AND_CR input flist, which contains the resource ID of the sub-balance to change or create. The consumption rule is set in the PIN_FLD_CONSUMPTION_RULE field of the **/balance_group** object of the various billing entities.

A non-currency resource without an explicitly set credit limit has a default credit limit of zero. If PCM_OP_BILL_SET_LIMIT_AND_CR is called for a nonexistent resource, the resource is created. In the specified balance group, the resource is created, the credit limit is set, and the current balance is set to zero.

PCM_OP_BILL_SET_LIMIT_AND_CR does not affect the current balances of the account.

To return all the fields in the event object, set the PCM_OPFLG_READ_RESULT flag. If this flag is not set, PCM_OP_BILL_SET_LIMIT_AND_CR returns only the POID.

To run PCM_OP_BILL_SET_LIMIT_AND_CR without changing data in the database, use the PCM_OPFLG_CALC_ONLY flag.

- If the flag is set, no fields in the database are changed and the event object is not created. The fields that would have been used to create the event object are returned to the caller.
- If the flag is *not* set, either the **/event/billing/limit** object or the **/event/billing/consumption_rule** object is created to record details of the operation.

## Changing Tax Exemption Information

A customer might be exempt from paying taxes levied by certain jurisdictions. For example, the customer might not be required to pay city or state taxes.

You record tax exemption information in the customer's account so BRM can send this information to the tax calculation software your company uses.

Use Customer Center to change tax exemption information. See the Customer Center Help.

You can customize the Customer Center properties file to change the exemption types offered. For example, you can override the **exemptionType.format** property to include a different tax exemption type. See "Adding a Tax Exemption Type" and "Configurator Properties Files" in *BRM Developer's Guide*.

## Adding or Changing a VAT Registration Number

You can record the customer's buyer value added tax (VAT) registration number in the customer's account. BRM sends this information to the tax calculation software your company uses.

Use Customer Center to change VAT registration numbers.

# Securing Billing Information with Data Masking

Customer billing information includes sensitive data such as banking details. BRM supports configurable masking of string type customer data fields in CM responses to clients and external systems and within logs. Use masking to ensure that sensitive customer billing information is protected.

See "About Securing Sensitive Customer Data with Masking" for information on configuring sensitive data masking.

# 10

# Managing Business Profiles

This chapter explains how to manage Oracle Communications Billing and Revenue Management (BRM) business profiles.

## About Business Profiles

*Business profiles* provide a way for BRM to validate bill units and other objects to determine how they are used. 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 (**/config/business_profile** object) includes the following:

- A set of requirements that a bill unit (**/billinfo** object) 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**

  See "About Validation Templates".
- 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, you check the value of this key in the bill unit's business profile. See "Getting Information about an Object's Business

Profile".

To create business profiles, see "Setting Up Business Profiles and Validation Templates".

**Note:** Customer Center does not support business profiles. You can, however, 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 more information on setting up business profiles and assigning bill units to them, see "Managing Business Profiles".

## About Validation Templates

To belong to a business profile, a bill unit and all its related objects, such as services, balance groups, and sharing groups, must conform to any requirements associated with the business profile. Requirements are specified in validation template (**/config/template** subclass objects) linked to the business profile.

A validation template includes the following:

- A set of iScript rules that determine whether a bill unit and its related objects meet the requirements of the business profile with which the template is associated. See "About Using iScripts to Validate Objects for Business Profiles".
- An array of key-value pairs used to retrieve information about the bill units that belong to the business profiles to which the validation template is linked. For example, if a service template linked to a business profile includes the key **IsServiceGold** with the value **Yes**, that indicates that bill units belonging to the business profile are associated with a premium ("Gold") service. To find out whether a bill unit is associated with a "Gold" service, you check the value of this key in the templates linked to the bill unit's business profile. See "Getting Information about an Object's Business Profile".

**Note:** Key-value pairs are not used in the validation process that determines whether a bill unit, balance group, or service object meets the requirements of a business profile.

A business profile can be associated with the validation templates listed below in Table 10–1:

***Table 10–1 Validation Templates for Business Profiles***

| Validation Template | Description |
|---|---|
| Bill unit validation template (**/config/template/billinfo** object) | Contains the rules used to validate every **/billinfo** object associated with the business profile. Only one bill unit validation template can be associated with a business profile. |
| Balance group validation template (**/config/template/balance_group** object) | Contains the rules used to validate every **/balance_group** object associated with the business profile. Only one balance group validation template can be associated with a business profile. |

*Table 10–1 (Cont.) Validation Templates for Business Profiles*

| Validation Template | Description |
|---|---|
| Group validation template<br>**/config/template/group** | Contains the rules used to validate every **/group** object associated with the business profile. Only one group validation template can be associated with a business profile. |
| Resource sharing group validation template<br>(**/config/template/group/sharing/*** objects) | Contains the rules used to validate every charge sharing, discount sharing, or profile sharing group associated with the business profile. Only one resource sharing group template can be associated with a business profile. |
| Ordered balance group validation template<br>(**/config/template/ordered_balgrp** object) | Contains the rules used to validate every **ordered_balgrp** object associated with the business profile. Only one ordered balance group template can be associated with a business profile. |
| Profile validation template<br>(**/config/template/profile** object) | Contains the rules used to validate every **/profile/acct_extrating** object or **/profile/serv_extrating** object associated with the business profile. Only one profile template can be associated with a business profile. |
| Purchased discount validation template<br>(**/config/template/purchased_discount** object) | Contains the rules used to validate every **/purchased_discount** object associated with the business profile. Only one resource sharing group template can be associated with a business profile. |
| Purchased product validation template<br>(**/config/template/purchased/product** object) | Contains the rules used to validate every **/purchased_product** object associated with the business profile. Only one resource sharing group template can be associated with a business profile. |
| Service validation templates<br>(**/config/template/service/*** objects) | Service objects are validated against the rules in the service template that most closely matches their service type. A business profile can be associated with only one template per base service (**/service**) and service type (**/service/***).<br>For example, if a business profile is associated with a **/config/template/service** template and a **/config/template/service/telco/gsm** template, **/service/ip** objects are validated against the former template and **/service/telco/gsm/data** objects are validated against the latter template. |

If a business profile is not associated with a validation template for a particular type of object, objects of that type are not validated because they do not have to meet any requirements to belong to the business profile.

To create validation templates, see "Setting Up Business Profiles and Validation Templates".

For more information on triggering business profile validation, see "About Using iScripts to Validate Objects for Business Profiles".

## About Using iScripts to Validate Objects for Business Profiles

Each validation template contains a set of iScript rules used to verify that an object conforms to the requirements of the business profiles to which its template is linked.

The iScript validation mechanism is triggered for a bill unit and *all* of its related objects whenever any of the following actions takes place:

- You assign a bill unit to a business group.

- You modify a bill unit that belongs to a business group.
- You create or modify an object related to a bill unit that belongs to a business group. For a list of valid objects, see "About Business Profiles".

BRM does not supply iScript validation rules because customer requirements vary. Instead, you must create custom rules to meet your business needs.

For more information on creating iScript validation rules, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

For more information on adding the rules to validation templates, see "Defining Validation Templates".

## About the Business Profile Configuration File

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 you installed the BRM software. 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".

## About Assigning Bill Units to Business Profiles

A bill unit (**/billinfo** object) 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. For a list of objects that have validation templates, see "About Business Profiles".

For more information, see "Assigning Bill Units to Business Profiles".

## 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
   ```

**Important:**

- 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**. See "Validating Your Business Profile Configuration File Edits".
- 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. Open the localization file *BRM_Home*/**sys/msgs/businessprofiles/business_profile_descr.***locale* in an XML editor or text editor.
6. Update the file with the changes you made to the **pin_business_profile.xml** file. Include the localized business profile name, description, and business profile ID number. For example:

```
DOMAIN = "Business profiles" ;
  STR
    ID = 1 ;
    VERSION = 1 ;
    STRING = "Convergent" ;
  END
  STR
    ID = 2 ;
    VERSION = 1 ;
    STRING = "Convergent Business Profile Description" ;
  END
```

7. Save and close the file.
8. Load the localized strings into BRM. See "Loading Localized or Customized Strings" in *BRM Developer's Guide*.
9. 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 more information on 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
- Defining Validation Templates
- Modifying Validation Templates
- 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").
- A description of the business profile ("Desc").
- A list of associated validation templates ("TemplateId" elements). For more information on how many validation templates can be associated with a business profile, see "About Validation Templates".
- A list of key-value pairs ("NameValue" elements). An unlimited number of key-value pairs can be associated with a business profile. See "About Business Profiles".
- An optional attribute ("type"). If the type attribute value is **Invoice**, the business profile is identified as an invoicing business profile. See "Specifying BI Publisher Invoice Report and Template Names in BRM" in *BRM System Administrator's Guide*.

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 10–2:

*Table 10–2 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.<br>Minimum length is 1 character.<br>Maximum length is 255 characters.<br>**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.<br>Maximum length is 255 characters.<br>**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:<br>■ The **name** value of a validation template defined in the file's **TemplateList** element.<br>■ The **type** value of the same template. | See "Template name" and "Template type". |

*Table 10–2 (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:<br>■ **key**: Character string used with **NameValue value** to describe characteristics of the bill units belonging to the business profile (for example, **IsSponsor**).<br>■ **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.<br>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">
```

**Important:** 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.

## Defining Validation Templates

In the business profile configuration file, validation templates (**/config/template** subclass objects) are defined as **Template** child elements of the **TemplateList** parent element. A **Template** child element consists of the following elements and attributes:

- The name of the validation template ("Template name").
- The type of object validated by the template's rules ("Template type").
- A description of the validation template ("Desc").
- The rules used to validate objects governed by the template ("Iscript").
- A list of key-value pairs ("NameValue" elements). An unlimited number of key-value pairs can be associated with a validation template. See "About Validation Templates".

The syntax of a **Template** child element is:

```
<TemplateList>
 <Template name="string" type="object_type">
  <Desc>string</Desc>
  <!-- Specify the Rules for object_type -->
  <Iscript />
  <!-- Specify List of Key Value -->
  <NameValue key="string" value="string" />
 </Template>
</TemplateList>
```

To create a validation template, add a **Template** child element to the **TemplateList** parent element. In the child element, specify values for the items listed in Table 10–3:

***Table 10–3 Template Elements and Values***

| XML Element or Attribute | Description | Possible Values |
|---|---|---|
| **Template name** | Character string used as the name of the validation template object (for example, **Prepaid balance group**). | Minimum length is 1 character.<br>Maximum length is 255 characters.<br>**Note:** This string is mapped to the PIN_FLD_NAME field in **/config/template** subclass objects. Values in this field can be used to populate a list of validation templates in a UI. When creating the string, take any UI length restrictions into consideration. |

*Table 10–3 (Cont.) Template Elements and Values*

| XML Element or Attribute | Description | Possible Values |
|---|---|---|
| **Template type** | The type of object the template validates. | Can be any of the following object types:<br>■ **/balance_group**<br>■ **/billinfo**<br>■ **/group/sharing**<br>■ **/group/sharing/***<br>■ **/ordered_balgrp**<br>■ **/profile**<br>■ **/profile/***<br>■ **/purchased_discount**<br>■ **/purchased_product**<br>■ **/service**<br>■ **/service/*** |
| **Desc** | Character string that describes characteristics of the objects governed by the template (for example, **Balance group for bill units with credit balances**). | Minimum length is 1 character.<br>Maximum length is 255 characters.<br>**Note:** This string is mapped to the PIN_FLD_DESCR field in **/config/template** subclass objects. Values in this field can be used to populate a list of validation template descriptions in a UI. When creating the string, take any UI length restrictions into consideration. |
| **Iscript** | Set of iScript rules used to validate objects governed by the template. | See "Using iScripts in Validation Templates". |
| **NameValue** | Key-value pair used to retrieve information about objects associated with the validation template. In each **NameValue** element, specify the following:<br>■ **key**: Character string used with **NameValue value** to describe characteristics of the bill units associated with the validation template (for example, **IsServiceGold**).<br>■ **value**: Character string used with **NameValue key** to describe characteristics of the bill units associated with the validation template (for example, **Yes** and **No**). | Minimum length is 1 character.<br>Maximum length is 255 characters. |

### Using iScripts in Validation Templates

BRM does not supply iScript rules for business profile validation. Instead, users must create their own rules. For more information on creating iScript validation rules, see "Creating iScripts and iRules" in *BRM Developer's Guide*.

After creating the rules, add them to the "Iscript" element of the appropriate **Template** element (see "Defining Validation Templates").

When the validation templates are loaded into the BRM database (see "Setting Up Business Profiles and Validation Templates"), the "load_pin_business_profile" utility compiles the iScript rules associated with each template and checks for errors. At startup, the Connection Manager (CM) compiles each iScript once, stores it in its cache, and maps it to the Portal object ID (POID) of the validation template object.

BRM calls the compiled iScript whenever business profile validation is triggered (see "About Using iScripts to Validate Objects for Business Profiles").

## Modifying Validation Templates

To modify a validation template already defined in your BRM system, redefine the validation template in your business profile configuration file (see "Defining Validation Templates"). You can change any aspect of a validation template except "Template 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 validation template.

## 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">
```

**Important:** 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.

### Validating Your Business Profile Configuration File Edits

After editing the **pin_business_profile.xml** file (see "Editing the Business Profile Configuration File"), you use the "load_pin_business_profile" utility to load the contents of the file into **/config/business_profile** objects and **/config/template** subclass objects.

Before loading the contents of the file, the utility validates the contents against the file's schema definition. If the contents do not conform to the schema definition, the load operation fails. The schema definition is in the *BRM_Home***/xsd/pin_business_profile.xsd** file.

The XML file is not directly linked to its schema definition file. Instead, it is linked to the *BRM_Home***/sys/data/config/business_configuration.xsd** *reference* file.

For more information on the XSD reference file, see "About Validating XML Configuration Files" in *BRM System Administrator's Guide*.

After validating the XML contents against the schema definition, the utility converts the XML file into an flist. From the flist's template array, the utility compiles the iScript rules. If any syntax errors occur, the load operation fails.

## Assigning Bill Units to Business Profiles

You can assign a bill unit (**/billinfo** object) to a business profile (**/config/business_profile** object) during or after account creation.

To assign a bill unit to a business profile during account creation:

1. Call the PCM_OP_CUST_COMMIT_CUSTOMER opcode.

   In the opcode's input flist, specify the **/config/business_profile** object's POID in the PIN_FLD_BUSINESS_PROFILE_OBJ field of the appropriate BILLINFO array element.

2. PCM_OP_CUST_COMMIT_CUSTOMER passes the business profile information to the PCM_OP_CUST_CREATE_ACCT opcode.

3. PCM_OP_CUST_CREATE_ACCOUNT calls the following opcodes to validate the **/billinfo** object and its balance groups and services against the rules in the validation templates associated with the business profile:

   - The PCM_OP_CUST_SET_BILLINFO opcode, which validates the **/billinfo** object against the bill unit validation template (**/config/template/billinfo** object) and sets the **/billinfo** object's PIN_FLD_OBJECT_CACHE_TYPE value.
   - The PCM_OP_CUST_SET_BAL_GRP opcode, which validates the balance group objects against the balance group validation template (**/config/template/balance_group** object) and sets the **/balance_group** object's PIN_FLD_OBJECT_CACHE_TYPE value. When a service is associated with a balance group, this opcode also validates the service object against the service validation template that corresponds to the service object type (**/config/template/service** or **/config/template/service/***) and sets the **/service** object's PIN_FLD_OBJECT_CACHE_TYPE value.

     **Note:** If the business profile is not associated with a validation template that matches a particular object type, validation is not performed on the corresponding object type (see "About Validation Templates" for details). Skipping validation for this reason is not equivalent to validation failure and does not prevent the bill unit from being assigned to the business profile.

4. One of the following results occurs:

   - *If no validation errors occur,* PCM_OP_CUST_CREATE_ACCOUNT calls PCM_OP_CUST_SET_BILLINFO to put the business profile POID in the PIN_FLD_BUSINESS_PROFILE_OBJ field of the new **/billinfo** object.
   - *If validation errors occur,* account and bill unit creation fail.

To assign a bill unit to a business profile after account creation, see "Changing a Bill Unit's Business Profile".

## Changing a Bill Unit's Business Profile

Use the following procedure to perform either of these operations:

- Assign a bill unit to a business profile after an account has been created.
- Change the business profile of a bill unit that currently belongs to a different business profile.

To change a bill unit's business profile, call the PCM_OP_CUST_CHANGE_BUSINESS_PROFILE opcode and include the following information in its input flist:

- The POID of the **/billinfo** object whose business profile you want to change.
- The POID of the new **/config/business_profile** object to assign the **/billinfo** object to.
- The POID of any associated **/balance_group** objects that must be modified for the new business profile *and* the required modifications.
- The POID of any associated **/service/*** objects that must be modified for the new business profile *and* the required modifications.

PCM_OP_CUST_CHANGE_BUSINESS_PROFILE performs these operations:

1. Validates the **/billinfo** object against the bill unit validation template (**/config/template/billinfo** object) associated with the new business profile and then does one of the following:

   If validation succeeds:

   a. Calls PCM_OP_CUST_SET_BILLINFO to change the **/config/business_profile** POID in the PIN_FLD_BUSINESS_PROFILE_OBJ field of the **/billinfo** object.

   b. If cache residency distinction is enabled, changes the **/account** object's PIN_FLD_OBJECT_CACHE_TYPE value if required, then validates all cache residency objects that are related to the account and are in the account context; for example, **/profile/acct_extrating** objects and account-level **/purchased_product** objects.

   If validation fails, returns an error.

2. Creates a list of all the objects associated with the bill unit.
3. For each balance group in the list, checks whether the input flist contains a requested modification.

   If a modification is requested, calls PCM_OP_CUST_SET_BAL_GRP to make the modifications and to validate the modified balance group object against the balance group validation template (**/config/template/balance_group** object) associated with the new business profile.

   If a modification is *not* requested, validates the balance group itself.
4. After validating a balance group, creates a list of all the services associated with the balance group.
5. If cache residency distinction is enabled, searches for objects related to each service in the list and validates them (for example, **/purchased_discount** and **/profile/serv_extrating** objects that belong to the account).
6. For each service in the list, checks whether the input flist contains a requested modification.

   If a modification is requested, calls the PCM_OP_CUST_MODIFY_SERVICE opcode to make the modifications and to validate the modified service object

against the service validation template that most closely corresponds to the service object type (**/config/template/service** or **/config/template/service/*** object) associated with the new business profile.

If a modification is *not* requested, validates the service itself.

7. Repeats steps 3 through 6 until all balance groups and services associated with the bill unit are validated.

   If any of the balance group or service validations fail, returns an error. If an error is returned, the business profile change initiated in step 1 does not take effect.

> **Note:** If the business profile is not associated with a validation template that matches a particular object type, validation is not performed on the corresponding object type (see "About Validation Templates" for details). Skipping validation for this reason is not equivalent to validation failure and does not prevent the bill unit from being assigned to the business profile.

For more information on cache residency, see "Configuring BRM for Cache Residency Distinction" in *BRM System Administrator's Guide*.

## Getting Information about an Object's Business Profile

A business profile (**/config/business_profile** object) and the validation templates (**/config/template** subclass objects) linked to it include key-value pairs in their PIN_FLD_PAIR arrays. To get information about the objects belonging to a business profile, you can retrieve the values of the keys in the business profile and its associated validation templates. For example, if a bill unit belongs to a business profile that has a PIN_FLD_PAIR_KEY field set to **IsPrepaidGold** and a corresponding PIN_FLD_PAIR_VALUE field set to **Yes**, you know that the charges associated with the bill unit are prepaid with a gold credit card.

To get the value of a key in the business profile and validation templates associated with an object, call the PCM_OP_CUST_GET_BUSINESS_PROFILE_INFO opcode. Include the following information in the opcode's input flist:

- The POID of the **/config/business_profile** object to which the related object belongs. These are the possible related objects:
  - **/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/***

- The key whose value you want the opcode to get.
- The type of object from which the key's value should be obtained. These are the possible object types:
  - **/config/business_profile**
  - **/config/template/balance_group**
  - **/config/template/billinfo**
  - **/config/template/group**
  - **/config/template/sharing**
  - **/config/template/ordered_balgrp**
  - **/config/template/profile/***
  - **/config/template/purchased_discount**
  - **/config/template/purchased_product**
  - **/config/template/service/***

For more information on key-value pairs, see "About Business Profiles" and "About Validation Templates".

# 11

# Managing Customers' Services and Products

This chapter describes how to manage customers' services and products; for example, adding Oracle Communications Billing and Revenue Management (BRM) services, giving discounts, and transitioning deals and plans.

For more information on creating products, deals, and plans, see "About Creating a Price List" in *BRM Setting Up Pricing and Rating*.

For more information on creating services, see "Adding Support for a New Service" in *BRM Developer's Guide*.

## Displaying Deal, Product, and Service Information

You can display status, discounts, purchase dates, and a history of events such as balance impacts.

You use Customer Center to display deal, product, and service information.

## Managing Services

A service is a capability that you provide to customers, such as telephony, Internet access, or email.

You add services to an account by purchasing plans and deals.

You can also change the service status and change how the services are provisioned; for example, add mail boxes to an email account.

### About BRM Service Names

BRM supports your services by using service subclasses. The service names that appear in Customer Center and Pricing Center use the names of the service subclasses; for example, **/service/ip** as show in Figure 11–1.

***Figure 11–1 Service Subclass Name Example***


### About Optional and Required Service Types

The PIN_FLD_TYPE value of a service determines whether the service is required or optional in an account. This value is dependent on the **validate_deal_dependencies** entry in *BRM_Home***/sys/cm/pin.conf**, the Connection Manager (CM) configuration

file. (*BRM_Home* is the directory in which you installed the BRM software.) This entry determines whether the deal associated with a service is required or optional.

- If the deal is required, PIN_FLD_TYPE is set to PIN_BILL_SERVICE_REQUIRED.
- If the deal is optional, PIN_FLD_TYPE is set to PIN_BILL_SERVICE_OPTIONAL.

When the PCM_OP_CUST_CREATE_CUSTOMER opcode creates the services in an account, it first validates whether the deal dependency functionality is enabled in the CM **pin.conf** file. If so, it sets the service's PIN_FLD_TYPE value accordingly.

### About Closing a Required Service

You cannot set the status of a required service to **Closed** by using Customer Center. Instead, you must call the PCM_OP_CUST_UPDATE_SERVICES opcode, which in turn calls PCM_OP_CUST_SET_STATUS opcode. Set the PIN_FLD_STATUS_FLAGS value in the input flist to PIN_STATUS_FLAG_DUE_TO_REQ_SRVC. This closes all services in the plan.

**Note:**

- You cannot activate an optional service that has been closed if the required service is closed.
- When you transition a deal or plan in Customer Center, the service associated with the old deal *is* closed, and its status flag value is set to PIN_STATUS_FLAG_DUE_TO_TRANSITION. You cannot change the status of a service with this status flag value.

For more information, see "Defining Deal Dependencies".

## Adding Services to an Account

To add services, customers must purchase a deal or a plan. Purchasing a deal adds the deal to an existing service in the account, whereas purchasing a plan adds the plan and a new service to the account.

Plans are typically in the **default-addon** plan list.

Customer service representatives (CSRs) can add services to customer accounts by using Customer Center.

Also, customers can add services themselves by using your customized Web pages.

## Activating and Inactivating Services

You change the customer's service status by using the **Service** tab in Customer Center. You can backdate service status changes to a date earlier than the current date. You can specify a date in the future on which the service status will change.

The status can be active, inactive, or closed.

- When a service is active, the customer can use the service.
- When a service is closed or inactive, the customer cannot use the service.

You can specify a date in the future on which the service status will change.

For more information on changing account and service status, see "Changing Account and Service Status".

You can edit the list of reasons why a service was activated or inactivated. See "Customizing the List of Reasons for Account Changes".

## Changing How Services Function

You use the service tabs in Customer Center to change how a service works. For example:

- Specify an IP address for a customer.
- Enter IP telephony speed-dial settings.

You use Customer Center to change service details.

## Controlling Service Usage

You control service usage in the following ways:

- Specifying how to authorize service usage. By default, a customer is not authorized to use a service if a credit limit has been reached.
- Inactivating services. When a service is inactivated, the customer cannot use it.
- Inactivating accounts. When an account is inactivated, no services in the customer's account can be used by the customer.
- Canceling products. When an account's product is canceled, the associated service cannot be used.

For more information on setting the account status, see "Changing Account and Service Status".

In addition to capturing usage information, BRM can also send information to the system that controls the service. For example, BRM can notify an IP gateway that a customer's credit limit has been reached and the call should be terminated.

## Managing GSM Service Provisioning

Use the following opcodes to manage provisioning:

- To publish a service order, use the PCM_OP_PROV_PUBLISH_SVC_ORDER opcode. This opcode sends an **/event/provisioning/service_order/***eventtype* event to the Provisioning Data Manager (DM), where *eventtype* is the type of event.

  This opcode takes as input the Portal object ID (POID) of the service order event and a substruct containing the service order information that should be sent to the Provisioning DM.

- To update a service order, use the PCM_OP_PROV_UPDATE_SVC_ORDER opcode. This opcode updates the status of an **/event/provisioning/service_order/*** event.

  An **/event/provisioning/service_order/*** event stores the service order and information such as the status, service order type, and actions required.

  When a response is received from a provisioning platform, PCM_OP_PROV_UPDATE_SVC_ORDER uses information in the input flist to update the status of an **/event/provisioning/service_order/*** event.

  PCM_OP_PROV_UPDATE_SVC_ORDER receives as input the POID of the service order, the service order status, and other service order information.

- To validate and modify parameters for updating service orders, use the PCM_OP_PROV_POL_UPDATE_SVC_ORDER policy opcode.

  This policy opcode is called by PCM_OP_PROV_UPDATE_SVC_ORDER when a response is received from a provisioning system.

  By default, the PCM_OP_PROV_POL_UPDATE_SVC_ORDER policy opcode validates and transforms a global system for mobile communications (GSM) service order. When a response is received from a provisioning system, this policy opcode maps the response parameters to corresponding fields in the **/event/provisioning/service_order/telco/gsm** object.

  The input flist to the PCM_OP_PROV_POL_UPDATE_SVC_ORDER policy opcode includes the complete response from the provisioning applications. Based on the type of the service order, you can modify or validate the response flist.

  By default, the PCM_OP_PROV_POL_UPDATE_SVC_ORDER policy opcode returns the input flist without any change, except in the case of GSM services where the input flist is validated and the format is modified to map the fields in the **event/provisioning/service_order/telco/gsm** object.

# Managing Products

A product is a pricing object that defines how to charge your customer for goods and services. A product consists of one or more rate plans that specify the cost of each billable event that affects a customer's account.

You use Customer Center to display each product that the account owns and the basic information about each product, including the product's service, purchase date, and status.

## Purchasing Products

Although you can manage products individually, you do not purchase individual products for a customer; instead, you purchase a deal in Customer Center that contains the product. See "Purchasing Deals".

Functionally speaking, the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode calls the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT opcode to purchase the products a deal contains.

### Handling Purchase, Cycle, and Usage Start and End Times

When using PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT, you pass the purchase, cycle, and usage start and end times. You can specify absolute dates, start on first usage and end relative to the start date, or start relative to the purchase date and end relative to the start date.

The start and end times are passed in by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. The start and end times are specified in the same way for PCM_OP_SUBSCRIPTION_PURCHASE_DEAL and PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT.

For more information on setting purchase, cycle, and usage start and end times, see "Managing Purchase, Cycle, and Usage Validity Periods of Products and Discounts".

If the system is configured for time stamp rounding, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT rounds the start and end times to 00:00:00 hours.

### About Delayed Cycle Start and End Times

If you configure delayed purchase, cycle, or usage start and end times when you set up your price list, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT sets the delayed start and end times in the **/purchased_product object** when the product is purchased.

The day of month to which the start date is set depends on whether you configure to align the start and end dates with the accounting cycle by setting the **fm_bill cycle_delay_align** entry to **1** in the CM **pin.conf** file.

For example, you configure a product to start relative to the purchase date, and you set the relative period in the pricing plan to one accounting cycle. If the account creation day is January 5, the billing day of month is 5, and the deal is purchased on January 20, then the start date is set as follows:

- If start and end dates are not aligned with the accounting cycle, the start date is set to February 20.
- If start and end dates are aligned with the accounting cycle, the start date is set to February 5 if the accounting cycle is short or March 5 if the accounting cycle is long.

> **Note:** If the purchase, cycle, or usage start and end time is modified in Customer Center while purchasing the deal or creating the account, any delay specified in Pricing Center is ignored.

### Applying Purchase and Cycle Fees to the Balance

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT handles the purchase and cycle fees for the product as follows:

- PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT does not apply either deferred purchase or cycle forward fees. These fees are applied when the **pin_cycle_fees** utility is run as part of the **pin_bill_day** billing script. See "Applying Deferred Product Purchase Fees".
- If the product has a purchase fee and if the purchase start time is not deferred, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT applies the purchase fee and creates the associated **/event/billing/product/fee/purchase** object.

> **Note:** Unlike flexible cycle forward events, BRM does not support custom events for purchase fee events. Purchase fees associated with a product are stored in **/event/billing/product/fee/purchase** object.

- If the product has a cycle forward fee and if the cycle start time is not deferred, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT applies the cycle forward fee by calling the necessary opcodes. This creates the **/event/billing/product/fee/cycle/cycle_forward**_feetype object, where feetype is the type of cycle forward fee.
- If the product is purchased as part of a deal transition, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT can waive the purchase fees based on the PIN_FLD_FLAGS value passed in by the PCM_OP_SUBSCRIPTION_TRANSITION_DEAL opcode. For more information, see "How Deals Are Transitioned".

- When PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT is called during product customization, the value of PIN_FLD_FEE_FLAG in the input flist determines whether cycle and purchase fees are applied:
  - **Cycle and purchase fees.** If PIN_FLD_FEE_FLAG is set to PIN_BOOLEAN_FALSE, cycle and purchase fees are not applied.
  - **Purchase fees.** If PIN_FLD_FEE_FLAG is set to PIN_BOOLEAN_TRUE, the opcode applies purchase fees. The opcode uses the purchase fee of the product (base or customized) that is valid at purchase time. Purchase fees are not applied for customized products created after the initial product purchase.
  - **Cycle fees.** If PIN_FLD_FEE_FLAG is set to PIN_BOOLEAN_TRUE, the opcode calls the appropriate opcode to calculate and apply cycle fees. Cycle fees are applied based on the validity of the base and customized products. See "Validity Periods and Cycle Fees for Customized Products".

  PCM_OP_SUBSCRIPTION_PURCHASE_DEAL sets the value of PIN_FLD_FEE_FLAG during deal purchase. See "How Deals Are Purchased".

### Applying Deferred Product Purchase Fees

To apply deferred purchase fees, use the PCM_OP_SUBSCRIPTION_PURCHASE_FEES opcode.

By default, purchase fees are applied at the time of product purchase. However, you can defer the purchase fees to a later date. For example, a customer can sign up for a product that is not available until a later date. The product's purchase fees are deferred and applied when the product becomes available.

PCM_OP_SUBSCRIPTION_PURCHASE_FEES is called by the **pin_cycle_fees** utility to apply deferred purchase fees. When **pin_cycle_fees** is run with the **-purchase** parameter, it searches for **/account** objects with an expired PURCHASE_START_TIME value. For each account that it finds, it calls PCM_OP_SUBSCRIPTION_PURCHASE_FEES to apply the purchase fees.

PCM_OP_SUBSCRIPTION_PURCHASE_FEES applies deferred purchase fees only for products with an expired purchase start time.

> **Note:** For item products, the **/purchased_product** object is removed after the deferred fees are applied.

After the purchase fees are applied to the account, an **/event/billing/purchase_fee** object is created for auditing purposes.

If the purchase fee is applied successfully, PCM_OP_SUBSCRIPTION_PURCHASE_FEES returns the POIDs of the **/account** object and the **/event/billing/purchase_fee** object.

### Enabling Product Purchases from Closed or Inactive Accounts

By default, closed and inactive accounts cannot purchase products. You may, however, want to make such purchases possible.

For example, you may want CSRs to create accounts that are closed, then purchase products for the account. This allows the customer to verify that the product is correct before charging the account. When the account is made active, the customer is charged.

When you purchase a product for inactive or closed accounts, the product's status also needs to be inactive. If the product's status is active, rating modules rate and apply charges for the product. If you do not want charges to be applied, change the product's status to inactive at the time of purchase or use Pricing Center to set the product's status to inactive in the deal. When the account is activated, change the product's status to active for charges to be applied.

To enable product purchases from closed or inactive accounts:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. In the **fm_bill** section, add the following entry:

   **fm_bill deal_purchase_for_closed_account**
3. Set the value of the entry to **1**.
4. Save and close the file.
5. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### How Products Are Purchased

To purchase a product, use PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT.

---

**Important:** Do not call PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT directly. This opcode is always called by PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

---

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT is called to purchase products for the account or service object specified in the input flist.

---

**Note:**

- If a **/service** object is specified, the product is purchased for that service and is a *service-level product*. The **/service** object must belong to the account.
- If the **/service** object is NULL, the product is purchased for the **/account** object and is an *account-level deal*.

---

PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT performs these operations:

1. If billing is due, triggers billing.
2. Opens a transaction.
3. Retrieves the product's package ID, name, and type, which are passed in the input flist, and opens a deal instance. For more information on how a deal is generated, see PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.
4. Validates that the product is available for purchase. A product is *not* available for purchase when any of the following apply:
   - The product's validity period has not yet started (the PURCHASE_START_T value is greater than the current time).
   - The product's validity period has already ended (the PURCHASE_END_T value is less than or equal to the current time).

- The cycle or usage start time is less than the purchase start time.
- The cycle or usage end time is greater than the purchase end time.
- There is a PIN_FLD_PERMITTED field for the product but the POID type is not explicitly permitted.
- The purchase quantity is **0** or is not passed.
- The purchase quantity is less than the minimum purchase quantity set in your price list.
- The purchase quantity is greater than the maximum purchase quantity set in your price list.
- An attempt is made to purchase a partial quantity.
- The total quantity purchased for this product exceeds the maximum ownership quantity in your price list for this account or service. This applies only to subscription products that have a maximum ownership quantity specified.
- The total quantity purchased for this product is less than the minimum ownership quantity in your price list for this account or service. This applies only to subscription products that have a minimum ownership quantity specified.

5. If the product purchase is backdated, validates that:
   - The date to which the product purchase is backdated is not prior to the G/L posting date.
   - The date to which the product purchase is backdated is not prior to the effective date of the account or service.
   - The date to which the product purchase is backdated is not prior to the date of the last status change of the account or service.

   See "About Backdated Product, Discount, Deal, or Plan Purchase".
6. If the **purchase product** provisioning tag is set for the product, calls the PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING opcode. See "Customizing Provisioning When a Product Is Purchased".

   If a **/config/provisioning_tag** object is associated with the product, this opcode runs the opcodes specified in that object to run at product purchase time. The opcode always runs the opcodes specified by the __DEFAULT__ provisioning tag. See "Using the Provisioning Tag Framework" in *BRM Setting Up Pricing and Rating*.
7. If the product is sponsored, sets PIN_FLD_SPONSOR_OBJ accordingly.
8. If the opcode is called for a customized product, adds the POID of the base product's **/purchased_product** object to the PIN_FLD_OVERRIDDEN_OBJ field.
9. Determines the various start and end dates for the product. See "Handling Purchase, Cycle, and Usage Start and End Times".
10. If called during advanced customization, verifies that the validity period of the customized product falls within the validity period of the base product:
    - The purchase start and end times of the customized product must fall within the purchase start and end times of the base product.
    - The purchase start and end times must not overlap with the purchase start and end times of any other customized product for the same base product.

- The usage start and end times of the customized product must fall within the usage start and end times of the base product.
- The cycle start and end times of the customized product must fall within the cycle start and end times of the base product.

11. Verifies the product purchase with the specified account.
12. Applies the product to the account:
    - If it is a subscription product, generates a **/purchased_product** object with a pointer to the **/account** object.
    - If it is an item product, creates an audit record for the **/au_purchased_product** object.

    **Note:** Item products cannot be deferred. The audit object is used for rerating events.

13. Sets the purchase, cycle, and usage discounts, if applicable.
14. Sets the product status in **/purchased_product**.
15. Checks the PIN_FLD_FLAGS value set by PCM_OP_SUBSCRIPTION_TRANSITION_DEAL to determine whether the product purchase is due to a deal or plan transition. If so, sets the value to PIN_TRANS_WAIVE_PURCHASE_FEES.
16. Applies the appropriate purchase and cycle fees to the balance. See "Applying Purchase and Cycle Fees to the Balance".
17. Updates the product flags for applying purchase and cycle fees.
18. Creates the audit log event object (**/event/billing/product/action/purchase**) after the product is added to the account and all applicable fees are applied.
19. Closes the transaction.

If the product purchase is successful, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT returns a confirmation message.

During a deal or plan transition, additional validations are performed that may prevent a successful purchase:

- If the service contains at least one required deal, the service flag is set to **Required**. If all deals for the service are optional, the service flag is set to **Optional**. A required service cannot be canceled; therefore, the purchase portion of the transition does not occur.
- If the system flag **validate_deal_dependencies** is **1**, BRM checks to see whether prerequisites exist or whether the deals are mutually exclusive before PCM_OP_SUBSCRIPTION_PURCHASE_DEAL is called.
- If the permitted field in the plan is **service_type**, the flag is set to **PBS**.

### Improving Product Purchase Performance

During product purchase, BRM retrieves product details from the rating cache by reading rate plans. However, reading a large number of rate plans for retrieving product details can consume a lot of time. This slows down product purchase.

You can improve the product purchase performance by enabling the retrieval of product details from the BRM database. For more information, see "Improving

Performance in Retrieving Purchased Offerings for a Bill Unit" in *BRM System Administrator's Guide*.

## Modifying Products

You use Customer Center to modify products in the following ways:

- Delay purchase fees.
- Add a comment.
- Change the quantity.

You can also customize the pricing of a product. See "Customizing Product Pricing".

You can modify products at the following times:

- During account creation.
- When adding a deal.
- During account maintenance.

### Changing Product Status

When creating an account or adding a deal, you can change the status of each product. For example, if using the product requires hardware setup, you can inactivate the product until setup is complete. You can then activate the product by using Customer Center. The customer is not billed for the product while it is inactivated.

---

**Note:** You cannot inactivate a product after it has been purchased. However, you can inactivate a service. See "Activating and Inactivating Services".

---

### Changing Product Quantity

You can change the quantity of a product, most often when it represents a one-time purchase and has no usage fees; for example, a product that gives five free hours as a sign-up bonus.

If the product's minimum quantity allows it, you can also decrease a product's quantity.

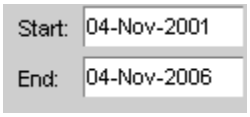
#### Applying discounts when you change product quantity

If you increase the quantity of a product that has a discount on purchase fees, that discount is applied to the entire purchase rather than to each unit of the product that is added to the account. For example, if you increase the quantity of a product in an account from one to five and the discount on purchase fees is $5, the customer receives a discount of $5, not $20.

### Setting Start and End Dates

You can set the start and end dates for the purchase, cycle forward fees, and usage rates. The start date sets the date when the customer's balance begins to be affected by the rate. The end date is the first date when the rate is no longer valid. In Figure 11–2, the last day that the rate is valid is November 3, 2006.

*Figure 11–2 Setting Start and End Dates*




You might want to delay a product purchase if the product depends on additional software or hardware setup that the customer has to wait for.

### Changing the Purchase, Usage, and Cycle Start and End Times

The PCM_OP_SUBSCRIPTION_SET_PRODINFO opcode is called to change the purchase, cycle, or usage start and end times of an account's purchased product. This opcode is called by the PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS opcode when the status of an account's product is changed.

After a product is purchased and its start and end dates have been set, you can change the start time to specify only another absolute date, and you can change the end time to specify only another absolute date or to end never.

> **Note:** If a product has been tailored to include one or more advanced customizations, be sure to consider the validity periods of the base product and all customized products when changing a validity period: The validity period of customized products must fall within the validity period of the base product, and customized products in the same deal cannot have overlapping validity periods. If these rules are not followed, PCM_OP_SUBSCRIPTION_SET_PRODINFO will not commit the product to the database.

To change the purchase, cycle, or usage validity period, specify the new start and end dates in their respective START_T and END_T fields in the PIN_FLD_PRODUCTS array in the input flist.

If you change the purchase, cycle, or usage start time from first usage to an absolute date, PCM_OP_SUBSCRIPTION_SET_PRODINFO generates an event that causes Pipeline Manager to update the validity period in the pipeline database.

The start and end dates are stored in the account's **/purchased_product** object.

If your system is configured to round time stamps, and if you modify the purchase, usage, or cycle start and end times, PCM_OP_SUBSCRIPTION_SET_PRODINFO rounds the new times to 00:00:00 hours.

BRM does not allow you to change a product's purchase, usage, and cycle start and end dates as follows:

- The product purchase start date if the purchase start date has elapsed.

  For example, on January 1, a product is purchased with the purchase start date set to January 15. On January 10, you can backdate the purchase start date to January 5. But, you cannot backdate the purchase start date on January 16; that is, you cannot backdate the purchase date when the January 15 date has passed.

- The product cycle start date if the cycle fees have been applied.
- The product purchase, usage, and cycle end date prior to the respective purchase start date.

- The product purchase, usage, cycle start or end dates prior to the G/L posting date. See "About Backdating Beyond the G/L Posting Date".

See "About Backdated Purchase, Usage, or Cycle End Dates".

### Calculating the Cycle Forward Fee

If you modify the cycle start time or cycle end time for the product in the middle of a cycle, PCM_OP_SUBSCRIPTION_SET_PRODINFO calculates the cycle forward fee or cycle arrears fee for the product.

If the product has a cycle forward fee, the PCM_OP_SUBSCRIPTION_CYCLE_FORWARD opcode is called to perform the following actions:

- If you change the cycle start time from a future time to the current time, applies the cycle forward fee for the current time through the end of the current cycle.
- If you change the cycle end time to an earlier time and the cycle forward fee is already applied for that cycle, refunds the cycle forward fee for the unused portion of the cycle.
- If you change the cycle end time to a later time and the cycle forward fee is already applied for that cycle, applies the cycle forward fee for the end of the old cycle end time through the end of the new cycle.

### Calculating the Cycle Arrears Fee

If you modify the cycle start time or cycle end time for the product in the middle of a cycle, PCM_OP_SUBSCRIPTION_SET_PRODINFO calculates the cycle forward fee or cycle arrears fee for the product.

**Caution:**

- Refunds for cycle arrears fees are not supported. Use caution when changing cycle end times.
- Changing the cycle end time for a cycle arrears fee has some limitations. See "Cycle Arrears Product Limitations".

If the product has a cycle arrears fee, the PCM_OP_SUBSCRIPTION_CYCLE_ARREARS opcode is called to perform the following actions:

- If you change the cycle end time from a future time to the current time, the cycle arrears fee is applied for the cycle.
- If you change the cycle end time from a future time to an earlier time, but later than the current time, the cycle arrears fee is *not* applied. However, the cycle arrears fee is applied during the next billing run.

### Cycle Arrears Product Limitations

- If you change the cycle end time *only once* in a cycle, the cycle fee is applied either the day of the change or the next billing day, depending on the new cycle end value. If the new PIN_FLD_CYCLE_END_T value is the same as the event time, the cycle arrears fee is charged at the same time; otherwise, it is charged at the next billing time.

**Note:** If you use time stamp rounding, the cycle end time is compared to the event date. Otherwise, the cycle end time is compared to the event time (the current time).

- If you change the cycle end date *more than once* in a cycle:
  - If the cycle arrears fee from the first change has not yet been applied, it will not apply after the second change is made.
  - If the new cycle end date is changed to **0** (never) or to a date later than the end of the current accounting cycle, cycle fees apply accordingly, depending on which of the following cancellation settings you specified when setting up proration for your pricing plan:

    **Charge full**: Applies no cycle fees for the whole cycle.

    **Based on usage and is proratable**: Applies cycle fees from the new cycle end date to the end of the cycle.

    **No Charge and proratable**: Applies cycle fees from the new cycle end date to the end of the cycle.

    **No Charge and is not proratable**: Applies full cycle fees.

### How Products Are Modified

To modify a product owned by a customer, use PCM_OP_SUBSCRIPTION_SET_PRODINFO.

PCM_OP_SUBSCRIPTION_SET_PRODINFO performs these operations:

1. If billing is due, triggers billing.
2. Opens a transaction.
3. Verifies the validity periods of customized products. See "About Customized Product Validity".
4. For backdating operations, validates and sets the product's purchase, cycle, or usage start and end dates to a backdated date. See "About Backdated Purchase, Usage, or Cycle End Dates".
5. Modifies the product information as specified. For more information on how PCM_OP_SUBSCRIPTION_SET_PRODINFO handles start and end dates, see "Changing the Purchase, Usage, and Cycle Start and End Times".
6. Creates an audit record object (**/au_purchased_product**) used for rerating events.
7. Applies the product changes to the product object (**/purchased_product**) owned by the account.
8. Calculates and applies appropriate cycle fees for the event to the balance and, if necessary, calls other opcodes to record the cycle events. See "Calculating the Cycle Forward Fee".
9. Creates the **/event/billing/product/action/modify** event objects.
10. Closes the transaction.

If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_SUBSCRIPTION_SET_PRODINFO returns the entire event flist for the events created as a result of the modification. Otherwise, it returns the event POID of all event objects created as a result of the modification.

An error occurs in the following cases:

- If an attempt is made to change the cycle start date to a date earlier than any period for which cycle forward or arrears fees have already been applied.
- If an attempt is made to change the cycle end date to a date before the current time but after the end of any period for which cycle forward or cycle arrears fees have already been applied.

## Managing Product Provisioning

To manage product provisioning, use the following provisioning policy opcodes:

- PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING. See "Customizing Provisioning When a Product Is Purchased".
- PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS. See "Getting a List of Provisioning Tags".
- PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING. See "Customizing Provisioning When Canceling a Product".

**Note:** The BRM provisioning tag framework is the preferred method of customizing provisioning when a product is purchased or canceled. See "Using the Provisioning Tag Framework" in *BRM Setting Up Pricing and Rating*.

### Customizing Provisioning When a Product Is Purchased

To set fields in a **/service** object at the time of purchase, use the PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING policy opcode.

This policy opcode is called by PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT when a product is purchased.

**Important:** Do not call this opcode directly.

- If a **/service** object is associated with a provisioning tag in the input flist, provisioning is invoked.
- If provisioning is invoked, it checks the tag table for the corresponding provisioning tag and executes the corresponding function call that modifies the appropriate fields in the **/service** object.

You can customize the provisioning functionality by:

- Changing the services, tags, and functions in the **provisioning_tags** array.
- Adding to or modifying the provisioning sub-functions in the **fm_subscription_pol_provisioning.c** file.
- Using Pricing Center to modify products associated with services by including the applicable provisioning tag.

### Getting a List of Provisioning Tags

To customize product provisioning, use the PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS policy opcode. This policy opcode retrieves data for provisioning from the **provisioning_tags** array.

This policy opcode is called by the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT opcode when a product is canceled.

> **Important:** Do not call this opcode directly.

Each service in the **provisioning_tags** array is associated with a provisioning tag and a function. The provisioning tag specifies the type of service, and the function determines what is to be done with that service. For example, the PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS policy opcode is called by Pricing Center to access the tag table and display the list of service types and their corresponding provisioning tags.

You can customize the provisioning functionality by:

- Changing the services, tags, and functions in the **provisioning_tags** array.
- Adding to or modifying the provisioning sub-functions in the **fm_subscription_pol_provisioning.c** file.
- Using Pricing Center to modify products associated with services by including the applicable provisioning tag.

### Customizing Provisioning When Canceling a Product

To customize provisioning when canceling a product, use the PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING policy opcode. This policy opcode clears fields in a **/service** object at the time of cancellation.

This policy opcode is called by PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT when a product is canceled.

> **Important:** Do not call this opcode directly.

- If a **/service** object is associated with a provisioning tag in the input flist, provisioning is invoked.
- If provisioning is invoked, it checks the tag table for the corresponding provisioning tag and executes the corresponding function call that clears the appropriate fields in the **/service** object.

## Upgrading Products

You can upgrade a customer's product by canceling the existing product and adding the upgraded product. For example, a customer can upgrade from a limited number of hours of Internet access to unlimited access.

> **Tip:** If your price list includes an upgrade path from limited hours to unlimited hours, you should include a rate plan in the upgrade product that removes free hours. That way, when a customer upgrades from limited hours to unlimited hours, the balance no longer includes free hours.

You can also upgrade products by including them in deals that you transition. For more information, see "Transitioning Deals".

## Canceling Products

You can cancel a customer's products by using the **Plans** tab **Actions** menu in Customer Center. You can cancel individual products or cancel all products in a deal at one time by canceling the deal; for example, to upgrade a customer to a new plan or service. See "Canceling Deals".

You can cancel a product immediately or backdate the cancellation to a date earlier than the current date. You can also set the cancellation to a future date. See "Setting Start and End Dates".

You can charge for product cancellations by creating a cancel rate for the product.

See "How BRM Cancels Base and Customized Products" for more information on canceling products that have been customized.

**Note:**

- In Customer Center, canceled products do not show up in the Product table by default. To display them, choose **Actions - Customize Product Table** on the **Product** tab.
- In the case of deferred product cancellation, the cancellation is applied when the **pin_bill_day** billing script is run.
- If the product is contained in a required deal, you must either transition the *plan* containing the required deal or close the service to cancel the product. See "About Plan Transitions in BRM".

### About Cycle Forward Fees and Product Cancellation

If a product's cycle forward fee allows proration, the customer is credited for any cycle forward fees that have been charged for but not used. BRM makes refunds based on the price list in effect at the time the product was purchased.

### About Free Resources and Product Cancellation

Typically, canceling a product does not remove free hours or minutes from a customer's balance. Upgrading a product typically does not remove them either. If a customer uses up more than the allowed free resources before canceling a product in the middle of a billing cycle, you can charge for the overuse of free resources. For more information, see "Charging for Overuse of Free Resources during Product Cancellation".

### Canceling Products by Closing the Account

When you close an account, all of the products it owns are canceled. See "About Activating, Inactivating, and Closing Accounts".

### Canceling Products without Charging a Cancel Fee

You can specify the amount of time after a purchase that a product can be canceled without charging the customer. For example, you might want to specify a product cancel tolerance of 30 minutes in case a CSR assigns the wrong product to an account and needs to cancel it without charging the customer.

To specify a cancellation tolerance:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the **cancel_tolerance** entry.

- **0** (Default) to always charge.
- Any other value entered in **cancel_tolerance** is in minutes.

For example (15 minutes):

```
- fm_bill cancel_tolerance 15
```

3. Save and close the file.
4. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### Canceling a Product

You can cancel a product immediately or backdate the cancellation to a date earlier than the current date. You can also cancel a product in the future by modifying the product end times. You use Customer Center to cancel products.

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

### Including Event Adjustments in Product Cancellation Refunds

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

> **Note:** You must install BRM 7.5 Patch Set 1 or later to use this business parameter.

To configure BRM to include event adjustments in product cancellation refunds, modify the **EventAdjustmentsDuringCancellation** parameter in the **subscription** instance of the **/config/business_params** object:

- When this parameter is enabled (**1**), BRM searches for any event adjustments associated with the product cycle fee and applies them to the refund amount.
- When this parameter is disabled (**0**), BRM does not include event adjustments when calculating product cancellation refunds.

To include event adjustments in product cancellation refunds:

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 full path as part of the file name.

2. Search the XML file for following line:

   ```
   <EventAdjustmentsDuringCancellation>0</EventAdjustmentsDuringCancellation>
   ```

3. Change **0** to **1**.

> **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 billing configuration.

4. Save and close the file.
5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for more information on how to use Object Browser.
7. Stop and restart the CM.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

### Charging for Overuse of Free Resources during Product Cancellation

When a customer cancels a product in the middle of a billing cycle, you can prorate free resources and charge for overuse. To prorate free resources, you set up the cycle grant rate plan and prorate the grant on cancellation. Then, you convert the non-currency resource balance to a currency resource balance by configuring a fold rate plan.

For more information on rate plans, proration, and fold events, see the following topics:

- "About Real-Time Rate Plans" in *BRM Setting Up Pricing and Rating*
- "About Proration" in *BRM Configuring and Running Billing*
- "About Fold Events" in *BRM Setting Up Pricing and Rating*

For example, consider a product that grants 400 free minutes for a cycle starting on January 1 and ending on January 30. Each free minute equals one currency resource unit. On January 15, if the customer cancels the product after using 300 free minutes, the free minutes grant is prorated to 200 and the overuse of 100 free minutes is charged to the customer's account by folding 100 free minutes to 100 currency resource units.

> **Important:** You can prorate free resources by using folds only if the relationship between the currency resource and the free resource is fixed. For example, one free minute equals one currency unit. It does *not* work if the conversion between the resources is based on tiered rates; for example, different rates for peak time and non-peak time.

#### Setting up products to charge for overuse of free resources

Use Pricing Center to perform these tasks:

1. Create a product for the free resource you want to offer to your customers. For example, free minutes.
2. Define a cycle forward rate plan that adds free resources to the product in every cycle.

   **Important:** To rate overuse of the free resource, make sure the credit limit of the free resource is unlimited.

3. Specify that the free resource is prorated based on usage when a customer cancels a product.
4. Define a cycle fold rate plan to fold the free resource.
5. Configure a fold to convert the free resource to the currency resource when there is overuse.

### Rating Delayed Events for Canceled Products

By default, BRM does not delete **/purchased_product** objects (product instances) when you cancel them. When BRM is set up to use delayed billing, BRM requires information about the canceled products to rate the delayed events that were generated before the product was canceled. To rate events for canceled products, the canceled product instances must persist as **/purchased_product** objects, which is the default behavior. To check your settings, see "Customizing Product Cancellation".

You can also choose to rate fold events for canceled products, including products in canceled deals. By default, fold events associated with canceled (inactive) products are not rated.

### Calculating Conditional Discounts When Canceling a Product

When canceling a product, discounts that are based on conditions cannot be reliably calculated. When a product 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 product 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 product 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 product 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 product 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 product, 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 product cancellation.

- If the discount calculated at product 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.

### Specifying to Delete Canceled Products

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

**Important:**

- Do not delete canceled products if you use delayed billing. See "Rating Delayed Events for Canceled Products".
- Do not delete canceled products if you rerate events. Events that use deleted products cannot be rerated.
- You cannot delete a product if provisioning tags are defined for that product. Products with provisioning tags are updated by the provisioning system and therefore must remain in the BRM database.
- Oracle recommends that you not delete canceled products and discounts because other external systems might also use them.

To automatically delete **/purchased_product** objects when a product is canceled:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the **keep_cancelled_products_or_discounts** entry.
   - **0** deletes the canceled **/purchased_product** objects.
   - **1** (default) keeps the deleted products in the **/purchased_product** objects.
3. Save and close the file.
4. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### How Products Are Canceled

To cancel a product, use PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT. This opcode cancels the products associated with the **/account** object specified in the input flist.

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT is called by the PCM_OP_SUBSCRIPTION_CANCEL_DEAL opcode to cancel each product associated with a specific deal. Depending on how many products are included in the deal, PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT may be called more than once.

The input flist identifies the products to cancel in the PIN_FLD_OFFERING_OBJ (**/purchased_product** POID) field. For more information on offering objects, see PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT performs the following tasks:

1. Closes billing if the account balance is affected.
2. Opens a transaction.
3. Verifies that the product is valid for cancellation.

4. Checks if the corresponding deal is a required deal. If so, does not perform the cancellation.
5. Calls the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL policy opcode to determine whether to cancel and delete the product, to cancel without deleting the product, or to not allow the cancellation. If the corresponding deal has any dependencies, the product cannot be deleted.

   **Note:** By default, BRM retains canceled products. You can change this behavior by customizing the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL policy opcode. See "Customizing Product Cancellation".

6. If the cancel product provisioning tag is set for the product, PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT calls the PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING policy opcode to clear fields in the service object. See "Customizing Provisioning When Canceling a Product".

   If a **/config/provisioning_tag** object is associated with the product, this opcode runs the opcodes specified in that object to run at product cancellation time. The opcode always runs the opcodes specified by the __DEFAULT__ provisioning tag. See "Using the Provisioning Tag Framework" in *BRM Setting Up Pricing and Rating*.
7. Validates the product quantity requested for cancellation. The quantity must be less than or equal to the quantity owned by the account.

   **Note:** When the quantity to cancel is less that the quantity owned, the product quantity is modified; the product is not canceled.

8. If the product cancellation is backdated, validates that:·
   - The date to which the product cancellation is backdated is not prior to the G/L posting date·
   - The date to which the product cancellation is backdated is not prior to the effective date of the account or service.·
   - The date to which the product cancellation is backdated is not prior to the last status change date of the account or service.

   See "About Backdated Product, Discount, or Deal Cancellation".
9. Sets the product's purchase, usage, and cycle end dates to the cancellation date.
10. Reads the PIN_FLD_FLAGS value passed in from PCM_OP_SUBSCRIPTION_TRANSITION_DEAL to determine if cancellation fees should be waived.
11. Cancels any customized products associated with the product.
12. For products that have not been customized, applies the following fees, if applicable:
    - If the product is active and has any cycle forward fee associated with it, and if the product is canceled in the middle of the cycle, the cycle forward fee is refunded for the unused portion of the cycle and an appropriate cycle forward event is generated.
    - If the product is active and has a cycle arrears fee associated with it, and if the product is canceled in the middle of the cycle, the cycle arrears fee is charged

for the used portion of the cycle and an **/event/billing/product/fee/cycle/cycle_arrears** event object is generated.

- If the product has a cancellation fee, the cancellation fee is applied and an **/event/billing/product/fee/cancel** event object is generated.

> **Note:** If the product's validity period is configured to start on first usage and has not yet been set, the product has not been activated. In this case, cycle fees are not charged.

13. For products that have been customized, applies cancellation fees based on whether a customized product is currently valid:
    - If a customized product is currently valid, the cancellation fees in the customized product are applied.
    - If no customized product is currently valid, the cancellation fees in the base product are applied.

    See "How BRM Cancels Base and Customized Products".
14. For customized products, refunds the customized product's cycle fees. Also refunds the cycle fees of the base product, then reapplies them based on the new validities. See "How BRM Cancels Base and Customized Products".
15. Creates the audit log **/event/billing/product/action/cancel** object to record the cancellation of the product.
16. Returns the POIDs of all event objects created as a result of the modification. For detailed information, see the output flist specification.

PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT fails in the following cases:

- If the specified product in the input flist does not exactly match the **/purchased_product** object.
- If the cancellation quantity is not passed in the input flist or is passed as **0**.
- If the product pricing quantity is less than the cancellation quantity specified in the input flist.

### Customizing Product Cancellation

To customize product cancellation, use the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL policy opcode.

By default, when a **product** is canceled, the **/purchased_product object** is not deleted, but its status is set to **Canceled**. In addition, the end date is set to the product cancellation date. The PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL policy opcode allows you to specify whether the **/purchased_product** object is deleted or remains with a status of **Canceled**.

> **Important:** BRM requires information stored in the **product** to rate events when you use delayed billing and when events are rerated. In these cases, **products** should not be deleted.

**Note:** You can also delete canceled products by setting a configuration parameter. See "Specifying to Delete Canceled Products".

You can customize the behavior of the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL policy opcode by setting the PIN_FLD_ACTION field to one of these values:

- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_ONLY, which sets the status of the **/purchased_product** object to **Canceled** and does not delete the **/purchased_product** object when canceled.
- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_DELETE, which deletes the **/purchased_product** object when it is canceled.
- PIN_BILL_CANCEL_PRODUCT_ACTION_DONOT_CANCEL, which stops the product cancellation.

**Note:** The **pin_bill.h** header file defines all action strings.

By default, the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL policy opcode reads the provisioning tag of the product object. If a provisioning tag is configured for the product, the action is set to PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_ONLY. If there is no provisioning tag, the action is set to PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_DELETE.

# Customizing Product Pricing

You can customize the pricing of a customer's products in three ways:

- Override the price for purchase and cycle events. See "Overriding a Product Price".
- Provide a fixed-amount or percentage discount that applies to the product as a whole. See "Providing Product Discounts".
- Modify rates or price models in the product by specifying positive or negative changes to resource balance impacts. See "Modifying Rates and Price Models in a Product".

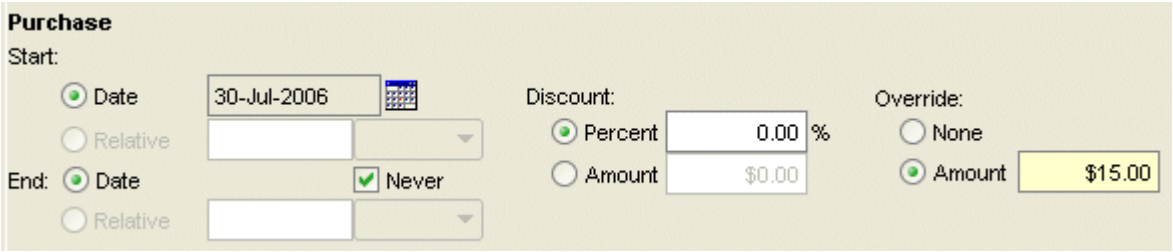
Price overrides and product discounts are collectively called *basic customizations*. Modifications to rates and price models in a product are called advanced customizations.

## Overriding a Product Price

You can override a product's purchase and cycle forward rates by entering a fixed price. This differs from a product discount, which is relative to an established price.

To override a product price, the CSR selects the product in the customer's account and enters an override price for the appropriate type of fee. Figure 11–3 shows Customer Center defining an override price of $15 for the purchase fee:

***Figure 11–3 Defining an Override***




**Note:**

- To make purchase or cycle events free, enter **0** as the override price.
- If you specify both a discount and an override price, the override price is discounted.
- To override a price when the account uses two currencies, make sure the display currency is the account's primary currency. Enter the override price in the primary currency.

**Important:**

- If a product has more than one cycle forward rate, do not enter an override price in the Cycle Info group. The override price substitutes for all cycle forward rates that affect the balance of a currency resource.
- Overriding the price applies only to scaled balance impacts. If the product includes a fixed balance impact, the override price is added to the price, it does not replace it.

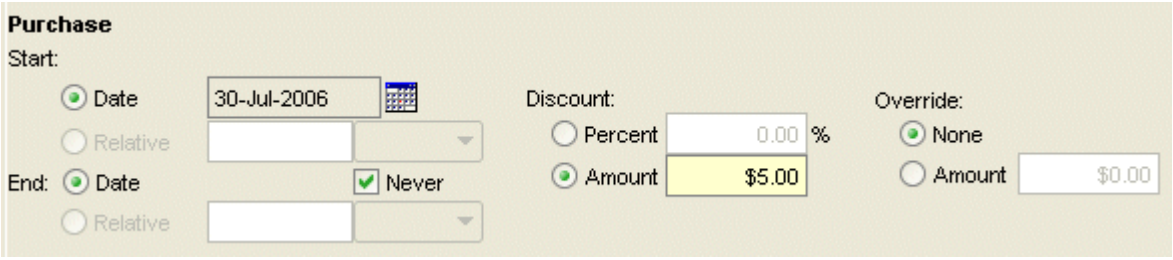
## Providing Product Discounts

You can provide percentage and fixed-amount discounts for a product's purchase and cycle forward rates and percentage discounts for usage rates. These types of discounts apply to the product as a whole.

**Note:** Product discounts are not the same as the separate, purchasable discounts that you can add to an account. See "About Discounts" in *BRM Configuring Pipeline Rating and Discounting*.

To discount a product, the CSR selects the product in the customer's account and enters the discount for the appropriate type of fee. Figure 11–4 shows Customer Center defining a discount of $5 for the purchase fee:

*Figure 11–4 Specifying a Discount*




**Note:** To give discounts when the account has two currencies, make sure the display currency is the account's primary currency. If you are giving a fixed-amount discount, enter the amount in the primary currency.

- You can use Pricing Center to specify if products are discountable. See "Providing Discounts with Deals" in *BRM Setting Up Pricing and Rating*.
- You can provide fractional discounts up to two decimal places; for example, 10.25%, but not 10.255%.
- You can configure the way discount results are calculated. See "Providing Discounts with Deals" in *BRM Setting Up Pricing and Rating*.

## Modifying Rates and Price Models in a Product

You can modify pricing for individual real-time rates and pipeline price models in a product. You can specify positive or negative percentage changes to resource balance impacts in the product's rates and price models. These types of modifications are called advanced customizations.

For example, you can increase the peak usage rate for a GSM customer by 10%, decrease the holiday usage rate by 50%, and increase the free minutes grant by 15%.

Advanced customization enables you to provide products that are configured specifically for a customer, without having to create new products in the product catalog.

When you modify a product in this way, BRM creates a customized products. Customized products do not appear in Pricing Center, but they are used during rating in place of the base product. See "How BRM Stores Customized Product Data".

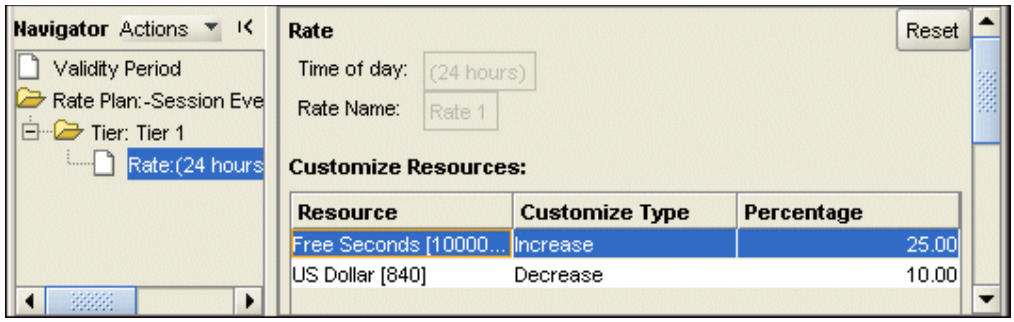
You set validity periods for customized products that determine the time periods during which they will be used. You can create multiple customizations of the same product with non-overlapping validity periods. See "About Customized Product Validity".

During rating, BRM checks for the existence and validity of customized versions of the product that apply to the rated event. If it finds a valid customized product, it uses the rates in that product rather than those in the base product. See "How BRM Rates Events with Customized Products".

You use the Advanced Customization dialog box in Customer Center to create customized products and to view their details after creation. The dialog box displays the hierarchy of real-time and pipeline pricing objects in a base product and enables you to view details about those objects. When you create a customized product, you specify its validity dates and percentage changes to resource impacts in rates and price

models as shown below in Figure 11–5.

***Figure 11–5 Customizing Products***




If you do not use Customer Center as your customer relations management (CRM) tool, you can call opcodes from another CRM tool to customize products. You can modify existing customizations with opcode calls. See "Using Opcodes to Customize Products" in *BRM Setting Up Pricing and Rating*.

You must grant permissions to Customer Center users to allow them to customize a product. You use the **accounttool/product/override** to control the ability to customize a product. This permission is not granted by default. You can define minimum and maximum customization limits.

A product can be customized only if the deal that contains it was created with the **Deal customization** setting at **Optional** or **Required**.

### How BRM Stores Customized Product Data

When you create an advanced customization of a product, BRM creates a unique **/product** object in the database specifically for that customization. This object stores not only the customized rates and validity periods but also uncustomized data from the base product. Both pipeline and real-time data is stored in the customized **/product** object. (BRM also creates customized data in the pipeline database and updates data modules in the pipeline. See "How BRM Stores and Updates Customized Pipeline Data").

BRM creates a separate **/product** object for each customization of the product. For example, if you customize a product to include a 25% discount for the month of June and later create a separate customization of the same product with a 10% discount for the month of August, BRM creates two separate customized **/product** objects.

When BRM creates a customized **/product** object, it calculates the new rates based on the percentage changes you entered. These new rates are stored in modified pricing components, such as price model steps for pipeline rate plans and balance impacts for real-time rate plans. Other pricing components included in the base product are copied in unmodified form to the customized product object.

A customized **/product** object includes all the pricing data from the base product:

- For real-time rate plans, customized objects include event maps, rate plans, rate plan selectors, rate tiers, day and date ranges, time ranges, rates, and balance impacts.
- For pipeline rate plans, customized objects include rate plans, rate plan versions, rate plan configurations, price model selectors, price models, and price model steps.

Because each customized **/product** object includes a copy of the uncustomized data in the original base product, the customized product is isolated from changes to the base product. For example, suppose you customize one rate in a product but leave all the other rates at their original values. BRM uses the rates in the customized product, including the ones you did not change, as long as the customized product is valid. Even if the base product changes during the customized product's validity period, BRM uses the original, unmodified rates along with the rate that you customized.

If you want changes to a base product to be reflected in a customized product, you must cancel the customized product and create a new customization based on the modified base product.

In some cases, a customized **/product** object may include multiple copies of a pipeline pricing component. This situation occurs because it is possible for multiple rate plan configurations and model selector rule sets to point to or customize the same price model. In this case, BRM includes a copy of the unmodified price model as well as a modified version of the same price model. For example, RatePlanConfigurationA might point to PriceModel1 in its original form whereas RatePlanConfigurationB *modifies* PriceModel1. The customized **/product** object would include a copy of the unmodified PriceModel1 as well as a copy that includes the modifications.

To distinguish them from other **/product** objects, customized **/product** objects include the field PIN_FLD_TAILORMADE with a value of **1**. BRM filters by this field to prevent customized products from appearing in Pricing Center. A customized **/product** object also includes the POID of the base product on which it is based, stored in the PIN_FLD_BASE_PRODUCT_OBJ field.

BRM stores customized real-time rates in the PIN_FLD_TAILORMADE_DATA field of the **/rate** object. This field stores resources and the percentage change to them as semicolon-delimited, comma-separated pairs. For example, reductions of 10% to the euro and US dollar resources would be stored in PIN_FLD_TAILORMADE_DATA as **EURO, -10;USD, -10**.

### How BRM Stores and Updates Customized Pipeline Data

If a customized product includes changes to a pipeline rate plan, BRM stores customized pricing data in the pipeline database as well as in the **/product** object. In addition, when you create or modify a customized product with a pipeline rate plan, BRM generates a notification event that causes pipeline modules to refresh their data caches.

When you customize a pipeline rate plan in a product, BRM creates copies of the relevant pipeline pricing components and stores them in the pipeline database. These customized pricing components are given a unique code based on the original component code. See "About Customized Product and Pipeline Pricing Component Names".

Pipeline Manager uses the data stored in the pipeline database to initialize pipeline modules if the pipeline is restarted after the product customization occurs. When customization occurs while the pipeline is active, modules use the event notification mechanism to update their data. See "Updating Pipeline Modules with Customized Data".

Table 11–1 lists the pipeline pricing components that can be customized and the database tables in which they are stored:

*Table 11–1 Pipeline Components That Can Be Customized*

| Pricing Component | Database Table |
|---|---|
| Rate Plan | IFW_RATEPLAN |
| Rate Plan Version | IFW_RATEPLAN_VER |
| Rate Plan Configuration | IFW_RATEPLAN_CNF |
| Price Model | IFW_PRICEMODEL |
| Model Selector | IFW_MODEL_SELECTOR |
| Model Selector Rule Set | IFW_SELECTOR_RULESET |
| Selector Rule | IFW_SELECTOR_RULE |
| Selector Rule Link | IFW_SELECTOR_RULE_LINK |
| Selector Detail | IFW_SELECTOR_DETAIL |

The database table for price models (IFW_PRICEMODEL) includes the TAILORMADE_DATA column, which stores customized resources and the percentage change to them as comma-separated pairs. For example, reductions of 10% to the euro and US dollar resources would be stored as **EURO, -10;USD, -10**.

The TAILORMADE column in the IFW_PRICEMODEL, IFW_MODEL_SELECTOR, IFW_SELECTOR_RULE, IFW_SELECTOR_DETAIL, and IFW_SELECTOR_RULESET tables indicates whether an object is customized. A value of **1** in the column indicates a customized object.

### Updating Pipeline Modules with Customized Data

When you create or modify a customized product, BRM generates a notification event: **/event/notification/price/tailormade_products/create** or **/event/notification/price/tailormade_products/modify**. These events include full customized product information, including pipeline rate plans, rate plan configurations, price models, and price model selectors.

The DAT_PriceModel, DAT_Rateplan, and DAT_ModelSelector modules respond to these events through their connection to the DAT_Listener module. They update their caches to include the customized data so that it can be used during rating.

### About Customized Product and Pipeline Pricing Component Names

BRM automatically creates the name of the customized **/product** object based on the name of the base product and the creation time. BRM prepends a hexadecimal version of the creation time, in seconds, to the base product name. For example, if the base product name is **GSMTelephony**, a customized product name could be **AE9C6890_GSMTelephony**. The original product name is truncated if necessary to maintain a size limit of 255 characters.

BRM also renames the copies it creates of pipeline pricing components. The code of each component is changed to include a two-letter abbreviation that indicates the component type, the hexadecimal creation time in milliseconds, and a unique five-digit integer value.

These are the abbreviations for the various component types:

- PR: pipeline rate plan
- PM: pipeline price model
- MS: pipeline model selector

- RS: pipeline model selector rule set
- SR: pipeline model selector rule
- SD: pipeline model selector detail

### About Customized Product Validity

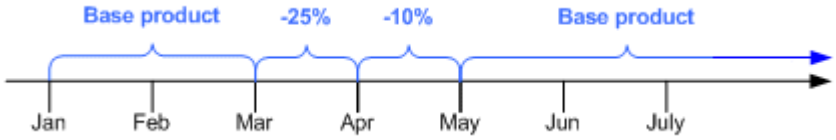
When you customize a product, you specify validity periods for the customization. You specify separate validity periods for changes to purchase, cycle, and usage fees. The purchase start and end times define the overall validity of the customization.

The validity periods of the customized and base products determine whether the base or customized product is used to rate usage events, apply purchase fees, and apply cycle fees. For example, suppose a product is valid starting January 1, 2006, with no ending date. You later customize the product to provide a 10% reduction on usage and cycle fees with validity from March 1, 2006, to April 30, 2006. The base product's cycle fees and usage rates apply in January and February. The 10% reduction on usage and cycle fees applies during March and April. The normal fees of the base product resume on May 1 and are valid until another customization takes effect as shown in Figure 11–6.

***Figure 11–6 Example Time-line for Product Fees***


You can create multiple customizations of a base product as long as their validities do not overlap. Figure 11–7 shows how you can customize a product to provide the subscriber with a 25% reduction on usage for March and a 10% reduction for April.

***Figure 11–7 Multiple Discounts Defined for a Product***




**Note:** You can create only one customized product for a base product under these circumstances:

- The base product validity is configured to start on first usage. The validity of the customized product must also start on first usage.
- The base product validity begins on a specific date and you set the customized product to start on first usage.

After the validity start date has been reached, you can create additional customizations for the base product.

You customize during product purchase and set the starting validity periods of the base and customized products to be the same.

The validity periods of a customized product must fall within the validity periods of its base product and cannot overlap with the validity periods of another customization of the same base product. Validity periods can fall in the middle of billing cycles and can be backdated as long as the dates fall within the validity of the base product.

**Note:** Backdating validity periods is allowed only when you first create a customization. You cannot backdate a customization after it has been created.

See "Validity Periods and Purchase Fees for Customized Products" and "Validity Periods and Usage Fees for Customized Products".

You can modify the validity periods of a customized product after its creation. For example, you can extend the length of a customization for an additional month.

**Note:** You cannot modify a customized product's validity period in such a way that it no longer falls within the validity of the base product or overlaps the validity of another customized product. Similarly, you cannot modify the validity of a base product in such a way that a customized product's validity no longer falls within the validity of the base product.

See the following for more information on how validity periods are used to calculate purchase, usage, and cycle fees:

- Validity Periods and Purchase Fees for Customized Products
- Validity Periods and Usage Fees for Customized Products
- Validity Periods and Cycle Fees for Customized Products

### Validity Periods and Purchase Fees for Customized Products

The purchase validity periods of the base and customized products at the time of purchase determine which is used to apply purchase fees.

If you customize a product at the time of the initial product purchase and set its purchase validity to begin at the same time as the base product, the purchase fee in the customized product is applied. If the purchase validity of the customized product begins after the initial product purchase, the base product's purchase fees are applied.

If you customize a product after the initial product purchase, the purchase fee of the customized product is not applied because the base product's purchase fee has already been applied. If you want to apply the customized product's purchase fee, you must trigger rerating for the purchase event. See "About Rerating" in *BRM Setting Up Pricing and Rating*.

### Validity Periods and Usage Fees for Customized Products

The usage validity of the base and customized products at the time of a usage event determine which product is used to rate the event. For example, if a customization has usage validity periods from August 1 to September 1, the usage rates defined in the customized product are used to rate events that occur during that month. Usage rates in the base product are used to rate all other events.

You may need to trigger rerating after customization to ensure that events that fall in the customized product's validity periods are rated properly. For example, suppose

that the GSM_Basic product rates peak calls at $0.10 per minute. A subscriber makes a 20-minute peak call on January 3, generating a usage event with a $2 balance impact.

If a CSR customizes this product on January 5 to reduce the peak usage rate by 25% with validity starting on January 1, you must rerate the January 3 call to ensure that the subscriber is charged correctly. After rerating, the balance impact will be $1.50, so an adjustment event of –$0.50 will be applied to the subscriber's balance. See "About Rerating" in *BRM Setting Up Pricing and Rating* for more information on rerating.

#### Validity Periods and Cycle Fees for Customized Products

The cycle validity periods of the customized and base products determine which product is used to apply cycle fees.

If a customized product exists at the beginning of the cycle and is valid throughout the cycle, its cycle fees are applied normally. This is also true in cases where a product is customized at purchase time with the customization's validity set to begin at the same time as the base product's. The cycle fees of the customized product are used until they are no longer valid.

If a customized product is valid for only part of a cycle, it contributes toward the total charge or refund based on the length of its validity. For example, suppose you discount the $10 monthly cycle fee of GSM_Basic by 15% with a validity period of March 1 to April 16. The cycle fee for March will be $8.50, reflecting the 15% discount for the whole month. The cycle fee for April will be $9.25: The cycle fee for the first half of the month is reduced by 15% to $4.25 and the full $5.00 fee is charged for the second half of the month.

If you customize a product's cycle forward fees in mid cycle, the already charged fees are refunded and new fees are calculated on the validities of the base and customized product. For example, suppose that on April 11 you customize a $10 monthly cycle forward fee by 10%, with validity from April 11 to May 1. BRM refunds the $10 cycle fee that was already paid. It then applies fees separately for the portion of the cycle covered by the base product ($3.33, or 10/30 of $10) and for the portion covered by the customized product ($6.00, or 20/30 of $10 minus 10%). The total cycle fee is therefore $9.33.

When the base product and one or more customized products are valid during a cycle, BRM creates separate cycle events to cover the validities of each product. Each event is rated separately based on the rates in the applicable product.

### How BRM Cancels Base and Customized Products

You cancel a base product with customized products the same way you cancel a product that has not been customized. See "Canceling Products". When you cancel a base product, any customized products associated with it are also canceled.

You cancel a customized product by using the **Advanced Customization** menu in the **Plans - Product Details** tab in Customer Center. Canceling a customized product does not cancel its base product.

> **Note:** You cannot cancel a customized product after the expiration of its validity.

#### Cancellation Fees

When you cancel a base product, BRM applies cancellation fees based on whether a customized product is valid or not:

- If a customized product is currently valid, the cancellation fees in the customized product are applied.
- If no customized product is currently valid, the cancellation fees in the base product are applied.

For example, suppose that the GSM_Basic product has a cancellation fee of $10. You create a customized version of the product with a 50% reduction of the cancellation fee, valid from January 1 to February 1. If the subscriber cancels on January 31, the cancellation results in a $5 balance impact (assuming that cancellation proration is set to **Charge for the entire cycle**). If the subscriber cancels on February 2, the balance impact is $10.

When you cancel a customized product without canceling its base product, no cancellation fees are applied.

#### Cycle fees

If you cancel a base product that has cancel proration set to **Calculate the charge based on the amount used**, the way cycle fees are prorated depends on whether a customized product is valid:

- If no customized product is valid, proration is based on the cycle fees in the base product.
- If a customized product is valid, proration is based on the cycle fees in the customized product.
- If you cancel a product during a cycle in which both base and customized products are valid, proration is based on the cycle fees and validity of both products. See "Calculating Prorated Cycle Fees and Refunds for Customized Products" in *BRM Configuring and Running Billing*.

If you cancel a customized product in mid cycle without canceling its base product, the cycle fees for the remainder of the cycle are first refunded based on the fees in the customized product. Cycle fees for the remainder of the cycle are then applied based on the base product's fees.

For example, suppose that the product GSM_Basic has a cycle forward fee of $10 and cancel proration is set to **Calculate the charge based on the amount used**. A CSR subsequently customizes this product for a subscriber to reduce the cycle forward fee by 20% with a validity period of January 1 to February 1.

If the customized product is canceled on January 16, BRM refunds $4. This amount is the customized product's prorated cycle fee for the unused period of January 16 to February 1. Because the base product is now valid, BRM then applies its prorated cycle fee of $5 for the same period. The total cycle fee is $9.

### How BRM Rates Events with Customized Products

BRM uses a customized product for rating instead of the base product from which it was derived, as long as the customized product is valid. For both real-time and pipeline rating, BRM checks to see whether a product has a customized version and then checks the validity of any customized products it finds. If a valid customized version is found, it is used for rating. For more information on validity, see "About Customized Product Validity".

There are some specific differences between the real-time and pipeline cases.

- In real-time rating, when BRM rates an event, it finds a list of products that apply to the event, filtering by parameters such as service and event time. It sorts the list of qualified products by priority and uses them in that order. BRM checks if any of

the qualified products have customizations and, if they do, whether the customizations are still valid. If a customized product is valid, it replaces its base product in the list of qualified products and it is used for rating. Customized products have the same priority as their base products.

You can control whether customized products are cached for use by the real-time rating engine. You enable and disable product caching in a **/config/business_params** entry. See "Enabling and Disabling the Caching of Customized Products" in *BRM System Administrator's Guide*.

- In pipeline rating, the FCT_Account module populates the CUSTOMER_DATA.PURCHASED_PRODUCT field of an event data record (EDR) with the account's purchased products that are valid for the time of the event. This list can include both base products and customized products. FCT_Account also populates the DETAIL.CUST_A.INTERN_RATING_PRODUCTS field with products to be used for rating, along with the priority of each product. The DAT_AccountBatch module then checks the base products in DETAIL.CUST_A.INTERN_RATING_PRODUCTS to determine whether customized versions exist. If DAT_AccountBatch finds a customized product for a base product and both products are valid, the base product is removed from DETAIL.CUST_A.INTERNAL_RATING_PRODUCTS. The customized product is therefore used for rating.

Because advanced customizations modify rates and price models, these changes are applied *before* charges are reduced by discounting. Charges that you reduce by customization can be further reduced by discounts. For example, suppose the GSM_Basic product charges $0.10 per minute for peak usage. A 20-minute peak call would result in a $2 balance impact. If you customize the product to reduce the peak usage rate by 25%, that same call results in a balance impact of $1.50. If the subscriber also owns a 10% discount on GSM usage, the balance impact would be further reduced by $0.15, resulting in a final balance impact of $1.35 for the call.

# Managing Discounts

Discounts are pricing objects similar to products; customers purchase discounts bundled in a deal.

## Purchasing Discounts

When purchasing a discount, you specify the following attributes:

- **Purchase quantity**

  The discount purchase quantity must be within the minimum and maximum allowed. See "Changing Discount Quantity".

- **Status**

  The discount status can be active or inactive. See "Setting Discount Status".

- **Start and end dates**

  The start and end dates specify when the discount begins and ends. See "Setting Discount Purchase, Cycle, and Usage Start and End Times".

For related opcodes, see "Subscription Management FM Standard Opcodes" in *BRM Developer's Reference*.

### About Purchasing Discounts in Mid Cycle

When a discount is purchased, BRM charges prorated and discounted cycle forward fees for the period during which the discount is valid. For more information, see "Prorating Cycle Fees after a Discount Purchase or Cancellation" in *BRM Configuring and Running Billing*.

### How Discounts Are Purchased

To purchase a discount, use the PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT opcode.

**Important:** Do not call PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT directly. This opcode is always called by the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode.

PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT performs these operations:

1. Closes the bill.
2. If billing is due, triggers billing.
3. Opens a transaction.
4. Reads the discount object information.
5. Performs the following validations:
   - Date validation, to make sure that the product is available for purchase.
   - Status validation, to check whether the discount is active or inactive.
   - Ownership quantity validation, to make sure that the discount instance purchase quantity is within minimum and maximum limits for ownership.
   - Purchase quantity validations, to make sure that a partial quantity purchase is not made and to check that the purchased quantity is within the minimum and maximum limits for purchase.
6. Calculates the start and end dates of the purchase/cycle/usage events based on either CSR customization or pricing object details. If passed relative dates, calculates absolute dates. If Content Manager is configured for time stamp rounding, all start and end dates round to 00:00:00 hours. For more information, see "Managing Purchase, Cycle, and Usage Validity Periods of Products and Discounts".
7. If the discount purchase is backdated, validates that:
   - The date to which the discount purchase is backdated is not prior to the G/L posting date.
   - The date to which the discount purchase is backdated is not prior to the effective date of the account or service.
   - The date to which the discount purchase is backdated is not prior to the date of the last status change of the account or service.

**Important:** If you backdate the purchase of a discount on a cycle forward or cycle arrears event to a previous accounting cycle, the discount will be applied only from the current accounting cycle.

See "About Backdated Product, Discount, Deal, or Plan Purchase".

8. If a **/config/provisioning_tag** object is associated with the discount, this opcode runs the opcodes specified in that object to run at discount purchase time. See "Using the Provisioning Tag Framework" in *BRM Setting Up Pricing and Rating*.
9. If applying discount validity rules, sets the cycle and usage start and end dates to one of the following:
   - The first day of the next accounting cycle (**no discount, valid from the middle of the cycle**).
   - The discount purchase date (**prorated discount, valid from the middle of the cycle**).
   - The first day of the current accounting cycle (**full discount, valid from the middle of the cycle**).
   - The purchase date of the discount (**Prorated discount, valid only part of the cycle**).
10. Determines whether the discount instance is to be created. If the opcode is not deferring item discounts or if the opcode has no associated purchase fee, creates a **/purchased_discount** object with a pointer to the **/account** object. For some item discounts, an instance is created, but it is removed after the deferred date when the purchase fee is applied.
11. If purchasing discounts in the middle of a cycle and the discount usage map is configured to support any cycle forward event types, refunds the discounted portion of the cycle forward fee as follows:
    a. Checks all discount usage maps to see whether the discount supports any cycle forward monthly event types.
    b. For each cycle forward event type supported by the discount, checks the charges arrays of all product instances for an array element of the cycle forward event type.
    c. Calculates the scale value for the portion of the cycle for which the discount is valid.
    d. Sends a request to rate normal prorated events and calculates the resulting prorated balance impact.
    e. Sends the event with an inverted flag set for recording.
12. Generates an audit log for the discount purchase, which sends a request to the PCM_OP_ACT_USAGE opcode to record an **/event/billing/discount/action/purchase** event.
13. Prepares the return flist.
14. Closes the transaction.

If the discount purchase is successful, PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT returns a message confirming the success.

## Validating Discount Dependencies

To validate discount dependencies, use the PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY opcode.

Configure mutually exclusive dependencies in the **/dependency** storable class.

PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY calls other standard opcodes to perform these operations:

- Validate the purchase time or the billing time if specified by a flag in the input. If the opcode validates the purchase time, any purchase time discounts are selected and validated against the **/dependency** object. If the opcode validates the billing time, any subscription, system, and shared discounts are selected and validated against the **/dependency** object.
- Validate discount-to-discount exclusion rules.
- Validate discount-to-price-plan exclusion rules, if appropriate.

If one of the operations fails, that operation, and only that operation, cancels. PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY returns a validation error and partially creates an account.

This opcode validates all the discount rules and returns the resulting discount values in the return flist if PIN_FLD_FLAGS does not get set to PIN_SUBS_FLG_RETURN_ON_FIRST_ERR in the PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY input flist.

This opcode returns an error message when the first error occurs if PIN_FLD_FLAGS is set to PIN_SUBS_FLG_RETURN_ON_FIRST_ERR in the PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY input flist. If the validation fails, this opcode returns a zero result, along with the POIDs of the discounts or plan and discount, whichever operation occurs first.

## Canceling Discounts

You typically cancel a discount instance when you cancel the deal in which it is bundled or when you close the account or service that owns the discount instance. You can also cancel a discount instance immediately or backdate the cancellation to a date earlier than the current date. A discount instance gets canceled automatically when the discount's purchase end date has been reached.

To cancel a discount instance, you must specify the quantity to cancel:

- If the quantity is the same as the quantity purchased by the account or service, the opcode cancels the discount instance.
- If the quantity is less than the quantity purchased by the account or service, the opcode does not cancel the discount; it subtracts that quantity from the internal count variable within the **/purchased_discount** object that represents the discount instance.

By default, when you cancel a discount instance, BRM retains the **/purchased_discount** object that describes the discount instance with a status of **Canceled**. You can specify whether to delete **/purchased_discount** objects or retain them with a status of **Canceled**. See "Specifying to Delete Canceled Discounts".

Changing the status of a discount instance to *canceled* sets the purchase, usage, and cycle end times to the cancellation time. See "Setting Discount Purchase, Cycle, and Usage Start and End Times".

For related opcodes, see "Subscription Management FM Standard Opcodes" in *BRM Developer's Reference*.

### Canceling Resource Sharing Group Owner Discounts

When you cancel a discount instance for an account or service that also owns a resource sharing group, you also cancel the discount instance from each resource sharing group that the account or service owns.

For more information on resource sharing groups, see "About Resource Sharing Groups" in *BRM Managing Accounts Receivable*.

### Rating Delayed Events for a Canceled Discount

By default, when you cancel a discount 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 instances. BRM uses this information to rate the delayed events that were generated before the discount was canceled. To rate events for canceled discounts, the canceled discount instances must not be deleted.

BRM does not delete canceled discounts by default. To ensure that canceled discount 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
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.

When you change the status of a discount to **Canceled**, you also set its purchase, usage, and cycle end dates to the cancellation date. See "Setting Discount Purchase, Cycle, and Usage Start and End Times".

### About Canceling a Discount in Mid Cycle

When you cancel a discount, you also prorate and charge cycle forward fees that may have been discounted for the period during which the discount is no longer valid. For more information, see "Prorating Cycle Fees after a Discount Purchase or Cancellation" in *BRM Configuring and Running Billing*.

### Specifying to Delete Canceled Discounts

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

**Important:**

- Do not delete canceled products if you use delayed billing. See "Rating Delayed Events for Canceled Products".
- Do not delete canceled products if you rerate events. Events that use deleted products cannot be rerated.

To delete **/purchased_discount** objects when you cancel a discount:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).

2. Change the value of the **keep_cancelled_products_or_discounts** entry.
   - **0** deletes **/purchased_discount** objects when you cancel the discount.
   - **1** (default) keeps the canceled discounts in **/purchased_discount** objects.
3. Save and close the file.
4. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### How Discounts Are Canceled

To cancel a discount instance, use the PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT opcode.

This opcode cancels the discounts for the **/account** object or **/service** object specified in the input flist.

> **Note:** If you specify a NULL **/service** object, you cancel discounts associated with the **/account** object. Otherwise, you cancel discounts associated with the **/service** object.

You identify discounts to cancel in the PIN_FLD_OFFERING_OBJ (**/purchased_discount** POID) field in the input flist. For more information on offering objects, see PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

To cancel a discount instance, PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT performs the following tasks:

1. Validates the discount quantity requested for cancellation. The quantity must be less than or equal to the quantity owned by the account or service. See "Changing Discount Quantity".

   > **Note:** When the quantity to cancel is less than the quantity owned, the discount quantity within the instance is decreased; the discount is not canceled.

2. If the discount cancellation is backdated, validates that:
   - The date to which the discount cancellation is backdated is not prior to the G/L posting date.
   - The date to which the discount cancellation is backdated is not prior to the account or service's effective date.
   - The date to which the discount cancellation is backdated is not prior to the date of the last status change of the account or service.

   > **Important:** If you backdate the purchase of a discount on a cycle forward or cycle arrears event to a previous accounting cycle, the discount will be applied only from the current accounting cycle.

   See "About Backdated Product, Discount, or Deal Cancellation".

3. If a **/config/provisioning_tag** object is associated with the discount, this opcode runs the opcodes specified in that object to run at discount cancellation time. See "Using the Provisioning Tag Framework" in *BRM Setting Up Pricing and Rating*.
4. Sets the purchase, cycle, and usage end dates for the **/purchased_discount** object to the cancellation date. See "Setting Discount Purchase, Cycle, and Usage Start and End Times".
5. If you set a discount validity rule, it sets the PIN_FLD_USAGE_END_T and PIN_FLD_CYCLE_END_T fields to one of the following:
   - The first day of the next accounting cycle.
   - The discount cancellation date.

   For more information on discount validity rules, see "About Applying Discounts Activated or Canceled in Mid-Cycle" in *BRM Configuring Pipeline Rating and Discounting*.
6. Calls the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT policy opcode and does one of the following:
   - Deletes the **/purchased_discount** object associated with the service or account.
   - Sets the status of the **/purchased_discount** instance to **Canceled** without deleting it.

   See "Customizing Discount Cancellation".

   **Note:** If the account is the parent of a discount sharing group, it also deletes, or cancels without deleting, the **/purchased_discount** object from the list of discounts shared by the account.

7. Applies cycle fees or refunds discounted cycle fee amounts. See "Prorating Cycle Fees after a Discount Purchase or Cancellation" in *BRM Configuring and Running Billing*.

   **Note:** If the discount's validity period is configured to start on first usage and has not yet been set, the discount has not been activated. In this case, cycle fees and refunds on discounted cycle fees are not applied.

8. Generates an **/event/billing/discount/action/cancel** event to record the cancellation.

PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT returns the POID of the **/event/billing/discount/action/cancel** event.

### Customizing Discount Cancellation

To customize how discounts are canceled, use the PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT policy opcode.

**Note:** You can also keep or delete canceled discounts by setting a configuration parameter. See "Specifying to Delete Canceled Discounts".

By default, when you cancel a **/purchased_discount** object, you set its status to **Canceled**. In addition, you set the end date to the discount cancellation date. The PCM_OP_SUBSCRIPTION_POL_SPEC_CANCEL_DISCOUNT policy opcode allows you to specify whether to delete the **/purchased_discount** object or retain it with a status of **Canceled**.

**Note:** When delayed usage events are expected to be loaded into BRM, BRM requires information stored in the **/purchased_discount** object to rate the delayed events. In this case, the **/purchased_discount** object should always be retained with a status of **Canceled and should not be deleted**.

You can customize the behavior of this policy opcode by setting the PIN_FLD_ACTION field to one of these values:

- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_ONLY, which sets the status of the **/purchased_discount** object to **Canceled** and does not delete it.
- PIN_BILL_CANCEL_PRODUCT_ACTION_CANCEL_DELETE, which deletes the **/purchased_discount** object.
- PIN_BILL_CANCEL_PRODUCT_ACTION_DONOT_CANCEL, which stops the cancellation of the discount.

**Note:** All these strings are defined in the **pin_bill.h** include file.

## Modifying Discount Attributes

You can modify the following discount attributes:

- Discount quantity. See "Changing Discount Quantity".
- Discount status. See "Setting Discount Status".
- Discount purchase, usage, and cycle start and end times. See "Setting Discount Purchase, Cycle, and Usage Start and End Times".

You use Customer Center to modify these discount attributes. See "Managing Discounts".

For related opcodes, see "Subscription Management FM Standard Opcodes" in *BRM Developer's Reference*.

### Changing Discount Quantity

You can change the quantity of a discount when you purchase or cancel it or when you purchase additional quantities of the same discount. Discounts can have minimum and maximum purchase and ownership quantity limits. The discount quantity purchased must be within these limits. For example, if a discount has a maximum quantity of 10, a subscriber can purchase 10 instances of the same discount. If a discount has a maximum quantity of 5, a subscriber can purchase 5 instances of the same discount.

For more information on setting up discounts, see "About Implementing Discounts" in *BRM Configuring Pipeline Rating and Discounting*.

### Setting Discount Status

The status of a discount can be **Active**, **Inactive** or **Canceled**. A discount is not valid when it is inactive. The discount does not apply to any user events generated while it is inactive.

When you change the status of a discount, you specify the new status and the reason for the status change.

Discount status can change in the following cases:

- When you purchase a discount: you can set the status to **Active** or **Inactive** in the case of deferred purchase.
- When you purchase an inactive discount: you can later reactivate it by setting its status to **Active**.
- When you cancel a discount: you set the discount status to **Canceled**. See "Canceling Discounts".
- When you change the status of an account or service that owns a discount: you change the status of the discount to the same status; when you close an account, you cancel any discounts it owns.

### Setting Discount Purchase, Cycle, and Usage Start and End Times

You specify a discount's validity period for the account that purchases the discount 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 discount rate. For more information, see "About Product and Discount Validity Periods" in *BRM Setting Up Pricing and Rating*.

You can modify a discount's purchase, usage, and cycle start and end times at the following times:

- When purchasing a discount.
- When canceling a discount. By default, the discount's purchase, cycle, and usage end times are set to the cancellation time. You can change the date the discount expires by modifying the discount's end time.
- When changing the status of a discount.

> **Important:** Do not change a discount'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's purchase, usage, or cycle end date prior to the discount's purchase start date or prior to the G/L posting date.

See "About Backdated Purchase, Usage, or Cycle End Dates".

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.

When setting the validity period during discount purchase, Customer Center calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. See "Managing Purchase, Cycle, and Usage Validity Periods of Products and Discounts".

When changing the validity periods of a purchased discount, Customer Center calls the PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO opcode. See "How Purchase, Cycle, and Usage Validity Is Modified".

### How Purchase, Cycle, and Usage Validity Is Modified

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO is called to change the start or end time of an account's discount purchase, cycle, or usage period. This opcode is called by the PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS opcode when the status of an account's discount is changed: for example, when you set a discount's status to **Inactive** when you purchase it and activate it at a later time.

After a discount is purchased and the start and end times are set, you can change the start and end times to specify only other absolute dates.

To change the discount's purchase, cycle, or usage validity period, specify the new start and end dates in purchase, cycle, and usage START_T and END_T fields in the PIN_FLD_DISCOUNTS array in the input flist. The cycle and usage start times must be on or later than the purchase start time, and the cycle and usage end times must be on or earlier than the purchase end time.

**Important:** Do not change a discount's purchase, cycle, or usage start time after the discount has been applied to the account; otherwise, the discount will be applied incorrectly.

If you change the purchase, cycle, or usage start time from first usage to an absolute date, the opcode generates an event that causes Pipeline Manager to update the validity period in the pipeline database.

The start and end times are stored in the account's **/purchased_discount** object.

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO performs the following tasks:

1. Resets the **/purchased_discount** object's purchase, cycle, and usage start and end times to the times specified in the input flist.

   If discount validity rules are set, PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO sets the start and end times accordingly. See "Implementing Discount Validity Rules" in *BRM Configuring Pipeline Rating and Discounting*.

2. For backdating operations, validates and sets the discount's purchase, cycle, and usage end dates to a backdated date. See "Setting Discount Purchase, Cycle, and Usage Start and End Times".

3. Applies cycle fees or refunds discounted cycle fee amounts. See "Prorating Cycle Fees When a Discount's Cycle Start or End Date Is Changed" in *BRM Configuring Pipeline Rating and Discounting*.

   **Note:** When only the purchase period is changed and the cycle period starts on first usage, if the validity of the cycle period has not been initialized by a first-usage event, this opcode does not apply cycle fees or refund discounted cycle fee amounts.

4. Generates an **/event/billing/discount/action/modify** event to record the modification.

PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO returns the POID of the **/event/billing/discount/action/modify** event.

For general information about modifying a discount in a deal, see "Modifying Discount Attributes".

### Customizing Snowball Discounts

A snowball discount distributes group discounts to the members of a discount sharing group.

You can customize how group discounts are distributed to members by using the PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT policy opcode. You call this policy opcode when you run billing and have discounts configured for billing-time discounting.

You can modify the PCM_OP_SUBSCRIPTION_POL_SNOWBALL_DISCOUNT policy opcode to specify an algorithm for distributing the total group discount grant to the individual group members. For instance, you can specify distribution of the group discount based on group member contribution.

## Managing Deals

A deal consists of one or more products and discount objects. By bundling products into a deal, you can provide discounts for those products. For example, you can create a deal that waives the sign-up fee if the customer registers before a certain date.

You can add or upgrade a customer's products by purchasing deals after you create the account. Customers can purchase a deal only if they have already purchased a plan that includes the service that the deal offers. For example, if a customer has already purchased a product that uses the email service, the customer can purchase another email deal. To purchase a deal that has a *different* service, you add the deal to another plan and then add the plan with the new service. See "Purchasing Deals".

You can associate a deal with a particular service or with any service owned by an account.

You can set up deal transitions that determine which deals can be purchased subsequently to another deal. See "Transitioning Deals".

You can set up deal dependencies that determine which deals you allow a customer to own. See "Defining Deal Dependencies".

You can customize account-level deals during plan transitions. See "Customizing Account-Level Deals during Plan Transitions" for more information.

### Purchasing Deals

You can add or upgrade products for customers by purchasing deals after you create the account. Customers can purchase a deal only if they have already purchased a plan that includes the service that the deal offers. For example, if a customer has already purchased a product that uses the email service, the customer can purchase another email deal. You can also backdate deal purchases.

You purchase deals for customers from the **Plans** tab of Customer Center. For more information on how deals are purchased, see "How Deals Are Purchased".

**Note:**

- If the customer upgrades to a deal that includes more free hours, cancel the old deal before purchasing the new deal. See "Canceling Products".
- When you purchase a deal, you can customize it by modifying one or more products in the deal. See "Modifying Products" and "Customizing Product Pricing".
- When you purchase a deal, you can backdate the purchase date for all of the products it contains by using the purchase wizard.
- You need the proper permissions to purchase or modify a deal. To modify a deal, the deal must be defined as customizable. You define this property in your price list.

You can set up dependencies between deals to specify:

- Deals that must be owned before another can be purchased.
- Deals that are required in a plan.
- Deals that cannot be owned at the same time.
- Deals that you can upgrade or downgrade to.

See "About Deal Dependencies" in *BRM Setting Up Pricing and Rating*.

### How Deals Are Purchased

To purchase a deal, use the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode. The products and discounts in the deal are purchased for the account or service object specified in the input flist.

**Note:**

- If you specify a **/service** object, you purchase the deal for that service as a service-level deal. The **/service** object must belong to the account.
- If you specify a NULL **/service** object, you purchase the deal for the **/account** object as an account-level deal.
- If multiple deals are purchased for the same service type, the value of the **/service** object's **PIN_FLD_SERVICE_ID** field identifies the service instance for the deal.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL creates an identifier for each deal purchase. This identifier is stored in the PIN_FLD_PACKAGE_ID field in the **/purchased_product** and **/purchased_discount** objects. The identifier distinguishes products and discounts bundled as part of a deal or a plan purchase.

**Note:** PIN_FLD_PACKAGE_ID is unique for every deal, except when deals are purchased in the same plan. In this case, all the offerings in the deals within the plan will have the same PACKAGE_ID. If you need to uniquely identify such a deal: to cancel it, for example, you need to provide the **/service** object along with the PACKAGE_ID. If the deals share the same service, you must also provide the **/deal** object.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL sets the purchase, cycle, and usage validity periods of the products and discounts in a deal. For more information, see "Managing Purchase, Cycle, and Usage Validity Periods of Products and Discounts".

If the deal's products or discounts grant resources that start on first usage (when the subscriber impacts the resource balance for the first time), you can specify that the validity period of all resources in the deal that start on first usage are set when one of those resources is impacted for the first time. To do this, set the PIN_FLD_GRANT_RESOURCES_AS_GROUP flag value in the PIN_FLD_FLAGS field.

If you set the deal for on-demand billing, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL helps to create the on-demand bill for purchase fees for products associated with the deal. You set up on-demand billing for deals and plans by using Pricing Center.

Before you purchase a deal, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls the PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY opcode to validate deal dependencies. If dependencies exist, the deal purchase cancels.

In addition, if you set a **validate_discount_dependency** entry in the **/config/business_params object,** PCM_OP_SUBSCRIPTION_PURCHASE_DEAL checks for a mutually exclusive relationship between any discounts packaged in the deal and any plan the account already owns. The opcode retrieves the list of subscribed discounts and price plans and it calls the PCM_OP_SUBSCRIPTION_VALIDATE_DISCOUNT_DEPENDENCY opcode to validate any mutual dependencies. If the opcode detects such a relationship, the deal purchase cancels. If no mutually exclusive relationship exists, the deal purchase continues. For more information, see "About Discount Exclusion Rules" in *BRM Configuring Pipeline Rating and Discounting*.

**Note:**

- Dependencies are not checked if the deal originated in an external customer relationship management (CRM) application. In that case, the input flist contains a type-only deal POID because no actual BRM deal exists.
- The opcode requires no validation if the PCM_OP_SUBSCRIPTION_CANCEL_DEAL opcode was called from the PCM_OP_CUST_COMMIT_CUSTOMER opcode or the PCM_OP_CUST_MODIFY_CUSTOMER opcode.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT opcode to create **/purchased_product** objects for each product in the deal. If a product is customized at the same time as it is created, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT is called multiple times to create the base product and its customized products.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL uses the PIN_FLD_FEE_FLAG field to control whether PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT applies fees. See "Applying Purchase and Cycle Fees to the Balance".

When the deal and all products and discounts for the deal have been purchased, the opcode records an **/event/billing/deal/purchase** event in the database for auditing purposes.

If you successfully purchase the deal, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL returns the **/account** POID and the POID of the **/event/billing/deal/purchase** event.

You do not purchase the deal in the following cases:

- If the **/service** object does not belong to the **/account** object.
- If the **/account** object or **/service** object is not valid.
- If status flags are not set for provisioning.

A closed account may purchase a deal if the CM configuration file has this entry set to **1**:

```
- cm deal_purchase_for_closed_account 1
```

### Managing Purchase, Cycle, and Usage Validity Periods of Products and Discounts

Purchase, cycle, and usage validity periods define when products and discounts are valid and when product fees are charged and discounted. For more information, see "About Product and Discount Validity Periods" in *BRM Setting Up Pricing and Rating*.

PCM_OP_SUBSCRIPTION_PURCHASE_DEAL calls PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT and the PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT opcode, which set the purchase, cycle, and usage validity periods of the deal's products and discounts, respectively.

If the system is configured for time stamp rounding, PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT and PCM_OP_SUBSCRIPTION_PURCHASE_DISCOUNT round the start and end times to 00:00:00 hours.

When called by Customer Center, PCM_OP_SUBSCRIPTION_PURCHASE_DEAL uses the purchase, cycle, and usage validity dates that are specified in the **/deal** object by default. If you specify alternative validity periods for products and discounts, those validity dates permanently override the dates specified in the **/deal** object.

> **Note:** Dates for advanced customizations do not permanently override the dates of the base product. When advanced customizations are not valid, the base product is selected for rating the customer's usage. For more information, see "Customizing Product Pricing".

To set the purchase, cycle, and usage start and end times, pass the following values in opcode input flists in the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays. In the following fields, the asterisk (*) indicates either PURCHASE, CYCLE, or USAGE.

> **Note:** The cycle and usage validity periods must fall entirely within the purchase validity period.

- Specify the start time as follows:

  - To start immediately (when the product or discount is purchased), set the value of PIN_FLD_*_START_T and PIN_FLD_*_START_UNIT to **0**. The opcode sets the start time to the purchase time.
  - To start on first usage (when the subscriber uses the product or discount for the first time) set the PIN_FLD_*_START_UNIT field to **-1** and the PIN_FLD_*_START_OFFSET field to **0**.
  - To start relative to the purchase time, set the relative unit (for example, days or cycles) in PIN_FLD_*_START_UNIT and set the number of relative units in PIN_FLD_*_START_OFFSET. The relative start time is calculated by adding the relative offset period to the purchase time.
- Specify the end time as follows:
  - To never end, set the value of PIN_FLD_*_END_T and PIN_FLD_*_END_UNIT to **0**.
  - To end relative to the start time, set the relative unit (for example, days or cycles) in PIN_FLD_*_END_UNIT and set the number of relative units in PIN_FLD_*_END_OFFSET. The relative end time is calculated by adding the relative offset period to the start time.

For more information on the unit and offset values, see "Specifying Purchase, Cycle, and Usage Start and End Times".

When the product or discount is purchased, the start and end times are calculated based on the purchase time. The validity period is set in the PIN_FLD_*_START_T and PIN_FLD_*_END_T fields of the account's **/purchased_product** and **/purchased_discount** objects.

If the validity period has a relative start or end time, the details about the relative offset are stored in the PIN_FLD_*_START_DETAILS and PIN_FLD_*_END_DETAILS fields of the purchased product or purchased discount. The details fields store three values: the mode of the validity period, the relative offset unit, and the number of offset units in the relative period. For more information, see "Storing Relative Start and End Times for an Account's Products and Discounts".

### Specifying Purchase, Cycle, and Usage Start and End Times

When the purchase, cycle, or usage period has an absolute start time (starts and ends on a specific date and time), the start and end times are passed in opcode flists in time stamp fields:

- PIN_FLD_PURCHASE_START_T
- PIN_FLD_PURCHASE_END_T
- PIN_FLD_CYCLE_START_T
- PIN_FLD_CYCLE_END_T
- PIN_FLD_USAGE_START_T
- PIN_FLD_USAGE_END_T

When the purchase, cycle, or usage period has a relative start or end time, the details about the relative period are passed in opcode flists in *unit* and *offset* fields:

- These unit fields specify the type of relative unit:
  - PIN_FLD_PURCHASE_START_UNIT
  - PIN_FLD_PURCHASE_END_UNIT

  - PIN_FLD_CYCLE_START_UNIT
  - PIN_FLD_CYCLE_END_UNIT
  - PIN_FLD_USAGE_START_UNIT
  - PIN_FLD_USAGE_END_UNIT

  The unit can be one of these values:

  - Seconds = 1
  - Minutes = 2
  - Hours = 3
  - Days = 4
  - Accounting cycles = 8
- These offset fields specify the number of units in the relative period:
  - PIN_FLD_PURCHASE_START_OFFSET
  - PIN_FLD_PURCHASE_END_OFFSET
  - PIN_FLD_CYCLE_START_OFFSET
  - PIN_FLD_CYCLE_END_OFFSET
  - PIN_FLD_USAGE_START_OFFSET
  - PIN_FLD_USAGE_END_OFFSET

The unit and offset fields are used in combination to determine the relative offset of the purchase, cycle, and usage periods. For example:

- If a product's PIN_FLD_CYCLE_START_UNIT has a value of **8** (accounting cycles) and PIN_FLD_CYCLE_START_OFFSET has a value of **3**, the cycle fee starts three accounting cycles after the product is purchased.
- If a discount's PIN_FLD_PURCHASE_END_UNIT has a value of **8** (accounting cycles) and PIN_FLD_PURCHASE_END_OFFSET has a value of **3**, the discount expires three accounting cycles after it is activated.

You can specify any number of units in the offset fields, up to 1048576. This is equivalent to approximately 12 days in seconds, 728 days in minutes, and 119 years in hours.

The relative offset information in the unit and offset fields is stored in the BRM database in *details* fields. For more information, see "Storing Relative Start and End Times for an Account's Products and Discounts".

#### About Backdated Purchase, Usage, or Cycle End Dates

Setting a product purchase, usage, or cycle start dates to a backdated date is different from backdating a purchase itself. When the product purchase, usage, or cycle start dates are set to a backdated date, though the cycle charges are applied from the backdated date, the change is effective only from the current time.

For example, you purchase a product on February 1, and set the PURCHASE_ START_ T, USAGE_START_T, and CYCLE_START_T fields of the product to a backdated date of January 15. In this case, the charges are applied from January 15, but the creation time of the product remains February 1 (current time). Thus, if usage events during the period of January 15 to February 1 are rerated, they are not rated with this product.

Here, if you enter January 15 as PIN_FLD _END_T in the opcode, the product becomes effective from January 15 (the creation time of the product is set to January 15), and the charges are also applied from the backdated date of January 15.

Oracle recommends that you use the latter procedure for backdating a purchase.

> **Note:** You cannot set the discount PURCHASE_ START_T, USAGE_ START_T, and CYCLE_START_T to a backdated date.

Similarly, backdating a product or discount's purchase, usage, or cycle end dates is different from backdating the cancellation. When the product or discount's purchase, usage, or cycle end dates are set to backdated date, though the cycle charges are refunded from the backdated date, the change is effective only from the current time.

For example, on February 1, you set the PURCHASE_ END_T, USAGE_END_T, or CYCLE_END_T fields of a product to the backdated date of January 15. In this case, the charges are refunded from January 15, but the change in the END_T to January 15 is effective only from the current time (February 1). Thus, if usage events during the period of January 15 to February 1 are rerated, they are rated with this product.

Here, if you enter January 15 as PIN_FLD _END_T in the opcode, the product is available for rating from January 15, and the charges are also refunded from the backdated date of January 15.

BRM does not allow you to backdate the following:

- The product purchase start date if the purchase start date has elapsed.

  For example, on January 1, a product is purchased with the purchase start date set to January 15. On January 10, you can backdate the purchase start date to January 5. But, you cannot backdate the purchase start date on January 16; that is, you cannot backdate the purchase date when the January 15 date has passed.

- The product cycle start date if the cycle start date has elapsed.
- The usage and cycle end date of a product prior to its purchase start date.
- The purchase, usage, and cycle end date of a discount prior to its purchase start date.
- The purchase, usage, or cycle start and end dates of a product prior to the G/L posting date. See "About Backdating Beyond the G/L Posting Date".
- The purchase, usage, and cycle end dates of a discount prior to the G/L posting date. See "About Backdating Beyond the G/L Posting Date".

#### Example of Backdated Purchase, Usage, and Cycle Start and End Dates

On April 1, consider that you create an account that owns a product with $9.95 as a cycle forward fee, 3600 as Anytime Minutes, and the start date set to June 20. On May 2, you change the start date to April 16. The charges will be as follows:

- $4.98 monthly charges and 1800 Anytime Minutes for the period from April 16 to May 1.
- $9.95 monthly charges and 3600 Anytime Minutes for the period from May 1 to June 1.

### Backdating Purchase, Usage, or Cycle Start and End Dates

To backdate the purchase, usage, or cycle start or end dates of a product or purchase, usage, or cycle end date of a discount during purchase, enter the backdated date in the respective date fields.

After a product is purchased, use the PCM_OP_SUBSCRIPTION_SET_PRODINFO opcode to backdate the purchase, usage, or cycle start and end dates. Similarly, use the PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO opcode to backdate the purchase, usage, or cycle end dates of the discount.

**Important:** If you set the end date of a discount on a cycle forward or cycle arrears event to a previous accounting cycle, the discount will be refunded only for the current accounting cycle.

### Storing Relative Start and End Times for an Account's Products and Discounts

Relative validity information in the PIN_FLD_*_UNIT and PIN_FLD_*_OFFSET fields that are passed in opcode flists are stored in the database in *details* fields:

- PIN_FLD_PURCHASE_START_DETAILS
- PIN_FLD_PURCHASE_END_DETAILS
- PIN_FLD_CYCLE_START_DETAILS
- PIN_FLD_CYCLE_END_DETAILS
- PIN_FLD_USAGE_START_DETAILS
- PIN_FLD_USAGE_END_DETAILS

When products and discounts are purchased, the details fields are stored in **/purchased_product** and **/purchased_discount** objects.

The start-details and end-details fields are 32-bit integer fields that store the mode of the validity period, the relative offset unit, and the number of offset units in the relative period:

- **Mode**: The mode specifies generally when the validity period starts or ends. The mode is stored in the lower 8th bit.

  When a product or discount is purchased, its start and end dates are set and the mode value in the start-details field is set to PIN_VALIDITY_ABSOLUTE (a value of **0**).

  The end-details field can have a value of PIN_VALIDITY_ABSOLUTE if the end date is specified or a value of PIN_VALIDITY_NEVER (a value of **2**) if the validity period never ends.

  **Note:** Before the product or discount is purchased, the mode in the details field of the product or discount in the **/deal** object can also specify other values such as *immediate* and *relative*. The mode helps to determine how the start and end times are set when the product or discount is purchased. For more information, see "Storing Start and End Times for Products and Discounts in a Deal" in *BRM Setting Up Pricing and Rating*.

- **Relative offset unit**: This value specifies the type of offset unit and corresponds to the value of PIN_FLD_*_START_UNIT or PIN_FLD_*_END_UNIT that is passed

in opcode flists. The relative offset unit is stored in the next four bits of the details field and can have one of these values:

- Seconds = 1
- Minutes = 2
- Hours = 3
- Days = 4
- Accounting cycles = 8

- **Number of offset units**: This value specifies how many offset units are in the relative period and corresponds to the value of PIN_FLD_*_START_OFFSET or PIN_FLD_*_END_OFFSET that is passed in opcode flists. The number of offset units is stored in the remaining 20 bits of the details field.

## Modifying Deals

You use Pricing Center to modify deals in the BRM database. Changes you make to deals in the database do not affect the deals already owned by customer accounts.

You can change the valid deal period or the products associated with the deal, and you can define prerequisites, mutually exclusive relationships, or transitions for deals. See the Pricing Center Help for more information.

You use Customer Center to modify deals owned by customer accounts; for example, to transition to a new deal.

### Validating Changes to Deals

To validate changes to deals, use the PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS opcode.

> **Note:** To validate deal-to-deal transitions, use the PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY opcode. See "Validating Deal Transitions".

PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS performs these operations:

1. Performs validation checks on all of the deals.
2. Calls the PCM_OP_SUBSCRIPTION_CANCEL_DEAL opcode to remove any canceled deals.
3. Removes the deals to be canceled from an account's list of deals.
4. Calls the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL opcode to add the new deals to the account's list of deals.
5. Applies proration rules to the products. If any cancellation or purchasing fees are defined, they are charged.
6. If a deal created by PCM_OP_SUBSCRIPTION_CHANGE_OPTIONS requires a new service, calls the PCM_OP_CUST_MODIFY_CUSTOMER opcode to create that service.
7. Creates an **/event/billing/product/action/cancel** object for the canceled deals.

### How Deals Are Modified

To modify a deal owned by a customer, use the PCM_OP_SUBSCRIPTION_CHANGE_DEAL opcode. This opcode changes the subscription products associated with a deal for an account.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL calls PCM_OP_SUBSCRIPTION_CANCEL_DEAL to cancel the subscription products associated with a deal and then calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL to purchase new subscription products and associates them with the deal.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL generates two notification event objects:

- The **/event/notification/deal/change** object signifies change progress. The information in the object indicates the canceled deal.
- The **/event/notification/deal/change_complete** object signifies change completion. The information in the object indicates the new purchased deal.

PCM_OP_SUBSCRIPTION_CHANGE_DEAL returns the POIDs of the event objects created by PCM_OP_SUBSCRIPTION_DEAL and PCM_OP_SUBSCRIPTION_PURCHASE_DEAL.

If cancellation or purchase transactions fail, PCM_OP_SUBSCRIPTION_CHANGE_DEAL rolls back any changes made. If any part of the transaction fails, the products associated with the account are not changed.

## Transitioning Deals

To transition deals, you use Pricing Center to set up transition rules, which are applied when you upgrade or downgrade a deal for a customer. This enables you to limit the deals that customers can switch to and still remain fully provisioned.

A deal cannot be transitioned under the following circumstances:

- The new deal is mutually exclusive to a deal currently in the account.
- The account is missing a necessary prerequisite deal.
- The deal is associated with an inactive service.
- *The service being transitioned from* and the service being transitioned *to* are not the same service.
- The deal is required in the associated plan.

### Validating Deal Transitions

To validate deal-to-deal dependency rules, use the PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY opcode.

Customer Center and the PCM_OP_CUST_SET_STATUS opcode call PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY.

PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY performs the following steps:

1. Confirms that the **validate_deal_dependencies** entry in the CM **pin.conf** file exists and, if so, performs validations; if not, it does nothing. See "Enabling Deal Dependencies Validation".
2. Checks for a **PIN_TRANS_NAME_DEAL_VALIDATION** entry in the input flist. If this entry exists, the validation checks have already been performed within the current transaction.

3. Verifies that the input is a list of deals. If **PIN_FLD_DELETED_FLAG** is set to **1**, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY performs the validations required if those deals were already deleted.

If the deals pass all validation checks, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY returns a PIN_FLD_RESULT value of **True**. If any deal violates any of the validation checks, this opcode fails and returns a PIN_FLD_RESULT value of **False**.

If the validation fails because the two deals are mutually exclusive, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY returns the POIDs of the two deals and a message indicating that the two deals are mutually exclusive.

If the validation fails because a prerequisite deal is missing, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY returns the POID of the deal missing a prerequisite.

### How Deals Are Transitioned

To transition a deal from one account to another, use the PCM_OP_SUBSCRIPTION_TRANSITION_DEAL opcode.

Customer Center calls this opcode to transition one deal to another in an account.

PCM_OP_SUBSCRIPTION_TRANSITION_DEAL performs the following steps:

1. Calls the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL policy opcode to perform any custom validation that you may have added. See "Customizing Deal Transitions".
2. Checks the **/dependency** and **/transition** objects to make sure that the deal transition does not violate any transition rules.
3. Checks the **/deal** object to make sure that the deal being transitioned is not a required deal in the associated plan. If it is required, the transition is not performed.
4. Performs the transition.
5. Charges cancellation fees for the old deal and purchase fees for the new deal based on these flags in the **/transition** object:
   - **0** charges fees.
   - **1** waives purchase fees.
   - **2** waives cancellation fees.
   - **3** waives purchase and cancellation fees.

   **Note:** All cycle fees and non currency resources are always prorated regardless of the product's cancellation proration setting.

   **Important:** The PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION opcode transitions valid services with a status of **Inactive**, but those services remain permanently inactive after the transition.
6. During the transition, calls the PCM_OP_BILL_CYCLE_FORWARD opcode and the PCM_OP_BILL_CYCLE_ARREARS opcode to prorate the cycle fees, if

necessary. If the PIN_TRANS_PRO_NORMAL_ON_TRANSITION flag is set, any new products prorate the last cycle fee and the first cycle fee. This flag overrides the user proration settings for product purchases and cancellations.

### Customizing Deal Transitions

You can customize how to transition a deal from one account to another:

- To get a list of deals and products available for transition, use the PCM_OP_CUST_POL_TRANSITION_DEALS policy opcode.

  Customer Center calls this policy opcode to return the list of deals available for an account to transition to and the products the deals contain. This policy opcode takes a deal POID and transition type (usually **upgrade** or **downgrade**) as input, reads the **/transition** object, and returns the list of deals available for transition, including their product names and descriptions.

  Use the PCM_OP_CUST_POL_TRANSITION_DEALS policy opcode to perform any additional filtering of deals before they are returned as available for transition. For example, use this policy opcode to limit certain deals to customers in a specific city.

- To validate how deals are transitioned, use the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode. This policy opcode allows customized validation based on account data during deal-to-deal transitions.

  For example, you may restrict a deal transition to customers from a particular location or require that customers own the first deal for a specific period of time before allowing the transition to a different deal.

  The PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL policy opcode is called by PCM_OP_SUBSCRIPTION_TRANSITION_DEAL before it performs any validation checks. By default, the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_DEAL policy opcode is an empty hook provided to facilitate customization. Usually, customization consists of transition rules based on account data.

**Note:** To customize how to transition a *plan* from one account to another, use the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode. See "Customizing Plan Transitions without Configuring /transition Objects".

## Defining Deal Dependencies

**Important:** Product Dependencies is an optional feature that requires a separate license.

### About Deal Dependencies

If you plan to support product dependency validations, you need to enable the deal dependencies validation to be able to check that no deal violates the prerequisites and mutual exclusivity rules. See "Enabling Deal Dependencies Validation" for more information.

You can define dependencies between deals in Pricing Center that set up the following relationships:

- **Prerequisites**. Specifies that an account must own a particular deal to be able to purchase an additional deal.

  **Note:** A prerequisite can contain deals of different services. For example, to own a GPRS deal, an account must own a GSM deal.

  **Note:** A prerequisite deal cannot contain item products. When you create a plan with alternate deals, and if the base deal contains an item product, the purchase fails.

- **Required deal**. Specifies whether deals are optional or required for plans. *Required* deals must be purchased when a plan is purchased, whereas *optional* deals can be added to an account at any time.
- **Mutual exclusivity**. Sets up a mutually exclusive relationship between two deals, so if an account owns one deal, it cannot own the other.
- **Allowed transitions**. Specifies which deals or plans can serve as replacements for others.

  Transitions specify the deals that customers can switch to and remain fully provisioned. While transitioning from one deal to another, your customers retain their devices, such as phone numbers and services.

  Customers owning deals associated with a primary service may transition to other deals associated with that primary service. The list of deals displayed as available for transition are all associated with a specific primary service.

### Enabling Deal Dependencies Validation

To enable the deal dependencies validation.

1. Open the CM configuration file *BRM_Home*/**sys/cm/pin.conf**.
2. *Uncomment* the **validate_deal_dependencies** entry to enable the deal dependencies validation:

   ```
   - fm_utils validate_deal_dependencies 1
   ```

3. Save and close the file.
4. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

See "How Deal Dependencies Are Validated" for more information.

### How Deal Dependencies Are Validated

You use PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY to validate deal dependencies. There are two ways to run PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY:

- Pass in an account and a list of deals.

  PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY checks whether owning the deals in the account and the deals you send would violate any dependency rules. You can also send in an account and a plan, and this opcode

performs the validation checks for the deals in the account and the deals in the plan.

- Pass in a list of deals to validate against another list of deals.

  PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY validates that owning both sets of deals would not violate any dependency rules.

In the first case, PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY accepts the POID for an account and validates that the deals in that account do not violate any deal dependency rules set in the **/dependency** object. You can also send in a **/plan** object, and PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY runs the validation checks between the deals in the **/account** and the deals in the plan.

If any deal violates a dependency relationship, the operation fails.

To validate two sets of deals, send in **/account -1** (without a POID); PCM_OP_SUBSCRIPTION_VALIDATE_DEAL_DEPENDENCY validates the deals. If any of the deals violates a dependency relationship, the operation fails.

## Deal Dependency Validations Involving Inactive or Canceled Products or Discounts

By default, BRM does not perform deal dependency validations for inactive or canceled products or discounts.

To enforce deal dependency validations for inactive or canceled products or discounts, you need to configure BRM to do the following:

- Maintain any deleted products in the **/purchased_product** objects to allow product-level and discount-level validations.

  To do so, set the value of the **keep_cancelled_products_or_discounts** entry to **1** in the **pin.conf** configuration file for Connection Manager. See "Specifying to Delete Canceled Products" 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. See "Enabling Deal Dependencies Validation" for more information.

- Perform deal dependency validations on inactive or canceled products and discounts.

  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**. See "Enabling Deal Dependency Validations for Inactive or Canceled Products and Discounts" for more information.

> **Note:** BRM performs the purchase time dependency validations between deals irrespective of the value of the **ProductLevelValidation** parameter.

### How BRM Acts on Deal Dependencies for Inactive or Canceled Products and Discounts

When BRM is appropriately configured to enforce deal dependency validations for inactive or canceled products or discounts and a deal A is the prerequisite of deal B:

- If deal B is owned by the account, BRM will not allow cancellation or inactivation of any products or discounts of deal A unless all the products or discounts of deal B are in canceled or inactive state.
- If any product or discount of deal A is in the canceled state, deal B cannot be purchased.
- If any product or discount of deal A is in the inactive state, deal B can be purchased only in an inactive state; that is, all products and discounts of deal B have to be purchased as inactive. All the products or discounts in prerequisite deal A should be first activated before activating the products or discounts in deal B.

**Note:** BRM does not perform validations between the validity dates of the prerequisite and dependent deals.

### Enabling Deal Dependency Validations for Inactive or Canceled Products and Discounts

Use the **pin_bus_params** utility to configure the **ProductLevelValidation** business parameter to enable deal dependency validations for inactive or canceled products and discounts. To do so:

1. Create an editable XML file from the **subscription** instance of the **/config/business_params** object by using the following command:

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

   BRM places 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 full path as part of the file name.

2. Locate the **ProductLevelValidation** entry in **bus_params_subscription.xml.out**.
3. Set the value of **ProductLevelValidation** as required. Your entry should be one of the following:
   - `<ProductLevelValidation>enabled</ProductLevelValidation>`
   - `<ProductLevelValidation>disabled</ProductLevelValidation>`
4. Save this updated file as **bus_params_subscription.xml**.
5. Load the modified XML file containing the business parameters for subscription into the appropriate **/config/business_params** object in the BRM database.

   ```
   pin_bus_params bus_params_subscription.xml
   ```

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

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

   For more information on reading objects by using the Object Browser, see *BRM Managing Customers*. For instructions on using the **testnap** utility, see *BRM Developer's Guide*.

7. Stop and restart Connection Manager. See the description for starting and stopping the BRM system in *BRM System Administrator's Guide*.
8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's*

*Guide*.

## Canceling Deals

You can cancel all the products in a deal in one operation, or you can cancel individual products. Canceling a deal is particularly useful when a customer wants to upgrade. You can cancel all of the products in the old deal as a group before purchasing the new deal for the customer's account. You can cancel a deal immediately or backdate the cancellation of a deal to a date earlier than the current date. You use Customer Center to cancel deals.

**Note:**

- After you cancel a deal, you cannot reactivate the deal or the products it contains.
- You cannot cancel a required deal. Instead, either transition to a new plan or close the service associated with the deal. See "About Plan Transitions in BRM".

### How Deals Are Canceled

To cancel a deal, use the PCM_OP_SUBSCRIPTION_CANCEL_DEAL opcode. This opcode cancels all products and discounts associated with the specific deal and then cancels the deal itself. If more than one instance of a deal has been purchased for the account or service, each deal instance must be canceled individually.

To cancel a deal, PCM_OP_SUBSCRIPTION_CANCEL_DEAL does the following:

1. Opens a transaction.
2. Checks if the deal is required within the corresponding plan. If so, the cancellation is not permitted.
3. From the input flist, retrieves the PIN_FLD_PACKAGE_ID, PIN_FLD_SERVICE_OBJ, name, and type of the products and discounts associated with the deal.

   If the deal being canceled has the same PIN_FLD_PACKAGE_ID and PIN_FLD_SERVICE_OBJ as another deal, then PIN_FLD_DEAL_OBJ is also required to uniquely identify the deal.
4. Validates that the deal is available for cancellation.
5. For each product, PCM_OP_SUBSCRIPTION_CANCEL_DEAL calls the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT opcode to cancel the product.

   The opcode does not directly delete customized products in the deal whose base products are also included in the deal. Customized products are automatically deleted when their base products are deleted by PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT. The opcode does delete customized products whose base products are *not* included in the deal.
6. For each discount, PCM_OP_SUBSCRIPTION_CANCEL_DEAL calls the PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT opcode to cancel the discount.
7. Creates an **/event/billing/deal/cancel** object for auditing purposes.
8. If the deal cancel is successful, returns the **/account** POID and the POID of the **/event/billing/deal/cancel** event.

# Managing Plans

You add new services to an account by purchasing a plan. Plans provide the deals, products, and discounts needed for configuring the new services.

You can also backdate plan purchases.

You also use plans to set and change credit limits in an account. For more information, see "Changing a Customer's Credit Limit".

BRM allows you to transition accounts from one plan to another.

## About Plan Transitions in BRM

You specify the rules that govern the transition of an account from the source plan to a target plan. BRM imposes certain limitations on when accounts can transition to or from other plans. It requires you to define plan transition rules for each plan-to-plan transition by manually configuring the transition rules in Pricing Center.

BRM uses the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode to transition accounts from the current (source) plan to a specified (target) plan. When a CSR transitions an account from one plan to another, Customer Center calls the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode to obtain a list of all the available plans to which the account can transition. The opcode then attempts the plan-to-plan transition.

You provide the required transition type as the value of PIN_FLD_TRANSITION_TYPE in the input flist for PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode. The transition type can be defined as one of the following:

- Not defined, set as **(0)**.
- Upgrade, set as **(1)**.
- Downgrade, set as **(2)**.
- Generation change, set as **(3)**. See "Defining a Generation Change for Plans".
- Custom transition type. See "Creating Custom Transition Types for Deals and Plans".

You can use the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode to customize plan transitions.

### Factors to Note about Plan Transitions in BRM

When you transition accounts from one plan to another plan in BRM:

- If the transition type is a generation change, the two plans do not have to share the same primary service. If the transition type is an upgrade or a downgrade, the two plans must share the same primary service.
- If the source and target plans are a valid combination, you can configure the transition rule such that the source plan retains the non-currency 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 deals (deals that are not part of the source plan) owned by the account. If all add-on deals are canceled, BRM closes the associated services and transitions the account to the target plan. If the add-on deals are *not* canceled, BRM returns an error and retains the source plan for the

account. Therefore, you must first cancel all add-on deals owned by the account *before* you transition the account to the target plan.

The following restrictions apply to plan transitions in BRM:

- You cannot backdate a plan-transition or generation change to a prior period.
- If you delete a service from a plan, 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 plan transition or a generation change. For example, if you have an account with ERA on friends and family and perform a plan transition or generation change, the ERA data is not transferred to the new plan.
- BRM, by default, retains the credit limits associated with the source plan for an account when the source and target plans for that account are associated with the same balance group but each plan 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 Plan Transition Rules in Pricing Center

You can manually configure the plan transition rule for each plan-to-plan transition in Pricing Center. For each such plan transition rule you configure in Pricing Center, BRM creates a **/transition** object to store the rules. For more information on how to define transition rules in Pricing Center, see the Pricing Center Help.

You can perform plan transitions to any plan without creating transition objects. This is a useful approach because for each plan in the BRM system, you need to specify all other plans to which the plan can transition. For example, if you have 300 plans and each plan can transition to or from any other plan, you define 89,401 (299 x 299) plan transition rules, thus creating 89,401 **/transition** objects.

In addition, every time you add a plan, you must define the transition rules for each existing plan to point to the new plan. As a result, the number of transition rules that you need to define in Pricing Center increases in proportion to any increase in the number of plans you support.

You can perform plan transitions to any plan without creating **/transition** objects to store the transition rules. See "About Providing Transition Rules Using Policy Opcode".

### About Providing Transition Rules Using Policy Opcode

You use the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode to automatically enable plan transitions to any plan without **/transition** objects.

---

**Note:** Customize the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode for only the plan transitions for which a **/transition** object is *not* defined in Pricing Center.

---

When BRM attempts to transition an account and the plan transition rule does not have a **/transition** object, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN uses the following information you provide in the *output* flist of the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode:

- Your specification on whether to charge any cancellation fees associated with the source plan for the account and purchase fees for the new product for this transition.
- The Primary Basic Service (PBS) that must apply to the plan transition.

> **Note:** If you do not provide these values, the opcode searches for a **/transition** object and if no **/transition** object exists for the transition rule, the transition will fail.

You can provide these values to handle some plan transitions and configure plan transition rules in Pricing Center for others. For more information on the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode, see *BRM Developer's Reference*.

> **Note:**
>
> - You can customize *plan* transitions in this way if the transition does *not* involve a generation change.
> - You cannot customize *deal* transitions in this manner. Pricing Center must be used to set up transition rules to transition deals.
> - You cannot transition plans under certain conditions. For example, you cannot transition a plan when add-on deals that are not part of the original plan are active and owned by the account.

### How BRM Transitions Accounts from Source Plans to Target Plans

The PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode transitions accounts from source plans to target plans in the following way:

1. Calls the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode to perform any custom validations.

   When the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode completes its actions, it passes the values provided by you to PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode.

2. For plan transitions where there is a **/transition** object for the transition rule, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN checks the **/dependency** and **/transition** objects to make sure that the transition does not violate any transition rules.

   For plan transitions where there is no **/transition** object for the transition rule, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN uses the value supplied by you in the output flist from PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode.

3. Validates the Primary Basic Service (PBS) that must be applied on the plan transition. For plan transitions where there is no **/transition** object for the transition rule, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN uses the value

supplied by you in the output flist from PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode. For:

- A regular transition, the two transition plans share a primary service. If not, the transaction fails.
- For a plan transition that involves a generation change, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN skips this step because the two plans do not have to share the same primary service.

4. Handles the non-currency grants in the source plan based on the value of PIN_FLD_FLAGS input field. If the value of FLD_FLAGS is PIN_SUBS_TRANSITION_CONTROL_ROLLOVER, the opcode calls a rollover function to ensure, if necessary, that the source plan retains non-currency grants and they are valid to the end of the cycle.

   To do so, the rollover function checks the plan list for the source and target plans and takes the following action:

   - If the plans are in the same plan list, the rollover function searches for all the sub-balances that are truncated because of the transition and extends the sub-balances to the original date before the truncation.
   - If the plans are *not* in the same plan list, this function searches for all the sub-balance buckets that are impacted by the transition and truncates the bucket to the current time.

5. Acts on the services in the plan-to-plan transition. If any services in the plans are listed as inactive, the transition is aborted.

   - If PIN_FLD_FROM_SERVICE is **NULL** or not specified in the flist, the new service in PIN_FLD_TO_SERVICE is added to the account.
   - If PIN_FLD_TO_SERVICE is **NULL** or not specified, the service specified in PIN_FLD_FROM_SERVICE is closed.
   - If PIN_FLD_FROM_SERVICE and PIN_FLD_TO_SERVICE have non-null values, only the deals are canceled and the deals from PIN_FLD_TO_SERVICE are purchased. Any account-level deals in FROM_PLAN are canceled. Any account-level deals in TO_PLAN are purchased.

6. For the plan being phased out due to a generation change in the transition, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN creates a **/schedule** object that closes the services associated with the plan at 00:00:00 hours at the end of the day of transition.

   Additionally, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN performs these steps:

   - Prepares the input flist for the PCM_OP_CUST_UPDATE_SERVICES opcode by specifying the value of END_T as 00:00:00 hours on the following day. For example, if the transition is issued on January 15, 2004, at 12:30:00, the plan being phased out ends at January 16, 2004, at 00:00:00 hours.
   - Calls the PCM_OP_ACT_SCHEDULE_CREATE opcode with the above input flist and the PCM_OP_CUST_MODIFY_CUSTOMER opcode number.
   - On the server side, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN:

     – Provides an option that customizes the end dates, allowing the end date to occur one day later.

– Allows a Product-End-Delay-For-Plan-Transition delay configuration, which is used if the end dates are not passed. This delay configuration is then applied to the date and time of transition as an offset value.

**Note:** ERA data is not transferred.

7. Charges any cancellation fees for the old deal and purchase fees for the new deal based on the PIN_FLD_FEE flag (in the **/transition** object or the custom value provided by you in the output flist from PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode).

**Note:** All cycle fees and non currency resources are always prorated regardless of the product's cancellation proration setting.

8. Creates **/event/notification/plan/transition** and **event/notification/plan/transition_complete** notification events. These events are not persistent.

BRM returns the following messages based on the success of failure of the transition:

- If the transition is successful, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN returns a message confirming the success.
- If the transition type is not a generation change and involves two plans that do not share a primary basic service, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN returns the message **The PBS does not match from-plan and to-plan**.
- If the transition violates any transition rules, PCM_OP_SUBSCRIPTION_TRANSITION_PLAN returns the message **Plan is not transitionable.**

### Customizing Plan Transitions without Configuring /transition Objects

You can customize plan transitions without configuring the **/transition** object by doing the following:

- **Obtain a list of plans available for transition**: Use the PCM_OP_CUST_POL_TRANSITION_PLANS policy opcode. Customer Center calls this policy opcode to return the list of plans available for transition.

  This policy opcode takes a plan POID and transition type (usually upgrade or downgrade) as input, reads the **/transition** object, and returns the list of plans available for transition as output. For more information, see PCM_OP_CUST_POL_TRANSITION_PLANS in *BRM Developer's Reference*.

- **Customize the validation as necessary**: This is optional. Use the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode to perform any additional validation checks or filters for transitioning plans.

  For example, you can restrict a plan transition to customers from a particular location or require that customers own the first plan for a specific period of time before allowing the transition to a different plan.

- **Provide custom values for service and fees**: This action is required if you plan to customize plan transitions *without* configuring the **/transition** object.

  Modify the *output* flist of the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode to specify the service that must be associated

with the plans and whether to waive purchase and cancellation fees associated with the plans. To do so:

1. Add the PIN_FLD_RESULTS array to the *output* flist of the PCM_OP_SUBSCRIPTION_POL_PRE_TRANSITION_PLAN policy opcode:

```
0 array RESULTS
1 int FEE_FLAG
1 str PERMITTED
```

2. Set the PIN_FLD_FEE_FLAG field in PIN_FLD_RESULTS to one of the values in Table 11–2:

***Table 11–2 PIN_FLD_FEE_FLAG Values***

| Value | Description |
|---|---|
| 0 | Charge cancellation fees for the old product and purchase fees for the new product. |
| 1 | Waive purchase fees. |
| 2 | Waive cancellation fees. |
| 3 | Waive purchase fees and cancellation fees. |

3. Set the PIN_FLD_PERMITTED field of this array to the Primary Basic Service (PBS) that must apply to the plan transition.

   For example,

```
0 PIN_FLD_RESULTS ARRAY [0] allocated 20, used 3
1 PIN_FLD_FEE_FLAG INT [0] 3
1 PIN_FLD_PERMITTED STR [0] "/service/telco/gsm/telephony"
```

### Customizing Account-Level Deals during Plan Transitions

You can customize account-level deals during plan transitions. To do so, provide the associated product information in the PIN_FLD_PRODUCTS input field for the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode. PIN_FLD_PRODUCTS is located in the PIN_FLD_DEAL_INFO substructure of the input flist for the PCM_OP_SUBSCRIPTION_TRANSITION_PLAN opcode.

For more information on PCM_OP_SUBSCRIPTION_TRANSITION_PLAN, see *BRM Developer's Reference*.

## Defining a Generation Change for Plans

A *generation change* allows you to transition your customers between 2G (second generation) and 3G (third generation) wireless plans and services. Plans 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. For example, the service types could be:

- **/service/telco/pdc**, a 2G service type based on the Personal Digital Cellular (PDC) standard for digital mobile telephony.
- **/service/telco/imt**, a 3G service type based on the International Mobile Telecommunications (IMT) standard for 3G wireless communications.

### How a Generation Change Works

Whether you are transitioning a customer from a 2G to a 3G plan or from a 3G to a 2G plan, both plans are valid all day on the day of transition. The plan that is being phased out expires at 00:00:00 hours at the end of the transition day; the plan being transitioned to becomes valid at 00:00:00 hours at the beginning of the transition day.

You set the transition rules between two plans. You can also set up standalone deals as add-ons and specify that the transition rules apply to those deals as well.

**Important:**

- When a transition between two plans is under way, no other transition is allowed between the two plans 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.

#### The New Plan

For the plan replacing the current plan, 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 deals associated with the plan are included in the transition.
- Any cycle fees for this plan 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 deals are purchased for the service.
- The new service is provisioned.

#### The Current Plan

For the plan 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 products 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 change the number of days the service being phased out is active, configure the **prod_end_offset_plan_transition** parameter in the **billing** instance of the **/config/business_params** object.

To modify the number of days the phased-out service is active:

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

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

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:

   ```
   <ProdEndOffsetPlanTransition>10</ProdEndOffsetPlanTransition>
   ```

3. Change **10** to the number of days you want the phased-out service to remain active (the default is 10 days). For example, if you change the value to **15**, the phased-out service remains active 15 days after generation of the new service. The minimum number of days you can keep a phased-out service active is **1** and the maximum is **31**.

   **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 billing configuration.

4. Save and close the file.

5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_billing.xml
   ```

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

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

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using **testnap**. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for more information on how to use Object Browser.

7. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

8. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

## 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, such as errors committed while performing subscription operations or errors resulting from a high latency between the networks and billing systems.

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

- A customer requests a product 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 product 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. See "About Backdated Account Creation".
- Backdated account and service status change. See "About Backdated Status Change".
- Backdated product, discount, deal, or plan purchase. See "About Backdated Product, Discount, Deal, or Plan Purchase".
- Backdated product, discount, or deal cancellation. See "About Backdated Product, Discount, or Deal Cancellation".
- Backdated product or discount purchase, usage, or cycle start and end date. See "About Backdated Purchase, Usage, or Cycle End Dates".
- Backdated sequential discount purchase, cancellation, and modification *if* the validity rules are set to **Prorated discount**. See the discussion of validity rules in *BRM Configuring Pipeline Rating and Discounting*.

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

- By using Customer Center.

  Subscription backdating is available only to customer service representatives (CSRs) who have the appropriate permission. Permissioning can be enforced only through Customer Center.

- By passing the backdated date in the PIN_FLD_END_T field of the corresponding subscription opcodes.

## 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, product, or discount is prior to the respective effective dates (creation dates).

## Rerating of Events for the Backdated Period

Backdated purchase or cancellation of a product or discount automatically triggers rerating. In these cases, BRM uses event notification to create rerate jobs that rerate the events in the product or discount. In all other subscription backdating scenarios, you must either run rerating manually or configure the trigger-dependent rerating. See "About Automatic Rerating of Backdated Events" in *BRM Setting Up Pricing and Rating*.

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

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

BRM lets you set up trigger-dependent rerating for the backdating operations. See "Setting Up Trigger-Dependent Rerating" in *BRM Setting Up Pricing and Rating*.

## About Backdated Product, Discount, Deal, or Plan Purchase

When you backdate a product, discount, deal, or plan purchase, BRM does the following:

- Sets the purchase, usage, and cycle start dates to the backdated date unless they are explicitly set to a different date.
- Prorates the cycle fee based on the proration settings.
- Applies the purchase fee on the backdate date.

See "Example of Backdated Product Purchase" and "Example of Backdated Discount Purchase".

BRM does not allow you to backdate the product, discount, deal, or plan purchase if:

- The backdated purchase date is prior to the G/L posting date. See "About Backdating Beyond the G/L Posting Date".
- The backdated purchase date is prior to the account or service's effective date. See "How Effective Time Is Used to Validate Backdating Operations".
- The backdated purchase 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 product purchase prior to December 15, which is the date of the last status change.

**Note:**

- BRM does not apply a fold, rollover product, or bill time discount for the past accounting cycles when you backdate the purchase of the fold, rollover product, or bill time discount.
- Backdating discounts with discount validity rules are supported only for discounts that are prorated. For more information On discount validity rules, see "About Applying Discounts Activated or Canceled in Mid-Cycle" in *BRM Configuring Pipeline Rating and Discounting*.

### How Sub-balance Buckets Are Created for Backdated Product Purchase

When the backdated product purchase spans beyond the current accounting cycle, multiple cycle events and sub-balance buckets are created. That is, if a backdated purchase of a product spans multiple cycles, one cycle event is created for each cycle and the free resource buckets are split into multiple cycles.

For example, consider that you create an account on January 1. On April 1, you backdate the purchase of a product to January 15. The product grants 3600 Anytime Minutes monthly. BRM creates the following three sub-balances for the three cycles:

- 1800 Anytime Minutes for January 15 to February 1.
- 3600 Anytime Minutes for February 1 to March 1.
- 3600 Anytime Minutes resources for March 1 to April 1.

**Important:** If you backdate the purchase of a discount on a cycle forward or cycle arrears event to a previous accounting cycle, the discount will be applied only from the current accounting cycle.

### How Cycle Fees Are Calculated for Backdated Purchase or Activation

BRM uses the PCM_OP_SUBSCRIPTION_CYCLE_FORWARD and PCM_OP_SUBSCRIPTION_CYCLE_ARREARS opcodes to calculate cycle forward and cycle arrears fees for the subscription operations.

When you backdate a purchase or activation, the opcodes determine the scale values that are used by real-time rating to prorate the cycle fee amount to be charged. The charges for the events that occurred during the backdated period are prorated and applied accordingly. These charges for the backdated period appear as normal cycle fees in the bill of the current cycle.

#### Example of Backdated Product Purchase

When you backdate the purchase of a product that includes cycle forward events, cycle forward fees are prorated and applied from the backdated purchase date through the current date.

For example, consider that you create an account on September 1 with the DOM set to the 1. On November 5, you backdate the purchase of a product that has the following details to September 15:

- Cycle forward charge of $9.95.
- Proration set to **Charge based on amount used**.

The following cycle charges are applied:

- $4.97 for the period from September 15 to October 1 (for 15 days).
- $9.95 for the period from October 1 to November 1.
- $9.95 for the period from November 1 to December 1.

These charges are included in the next bill that gets generated during the bill run on December 1. This bill also includes the charges that are generated for the period of December 1 to January 1.

In the same example, if the product being purchased has cycle arrears fees instead of cycle forward fees, the two cycles until the current cycle are charged as follows:

- $4.97 for the period from September 15 to October 1 (for 15 days).
- $9.95 for the period from October 1 to November 1.

These charges will get included in the next bill that gets generated during the bill run on December 1.

#### Example of Backdated Discount Purchase

Consider that on April 1, you create an account that owns a product with a cycle forward fee of $50. The cycle fee of $50 for April1 1 to May 1 is applied. On April 16, purchase a discount that applies 10% on the monthly cycle fees, backdated to April 1. A discount of $5 is applied for the period of April 1 to May 1.

### Backdating a Product, Discount, Deal or Plan Purchase

To backdate a product, discount, deal, or plan purchase, enter the backdated time in the PIN_FLD_ END_T field of the corresponding opcode. For example, to backdate a product purchase, enter the date to which you want to backdate the purchase in the PIN_FLD_ END_T field of the PCM_OP_SUBSCRIPTION_PURCHASE_PRODUCT opcode.

## About Backdated Product, Discount, or Deal Cancellation

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

**Note:** If you backdate a discount purchase to a date prior to the product 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.

**Important:** If you backdate the purchase of a discount on a cycle forward or cycle arrears event to a previous accounting cycle, the discount will be refunded only for the current accounting cycle.

BRM does not allow you to backdate the product, discount, or deal 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 product or discount's effective date. For example, a product or discount 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 product cancellation prior to December 15, which is the date of the last status change.

**Note:**

- When you backdate the cancellation of a fold, rollover product, or bill time discount, you must run rerating to apply the corrections on the events that occurred after the backdated date.
- Backdating discounts with discount validity rules are supported only for discounts that are prorated. For more information on discount validity rules, see "About Applying Discounts Activated or Canceled in Mid-Cycle" in *BRM Configuring Pipeline Rating and Discounting*.

### How Cycle Fees Are Calculated for Backdated Cancellation or Inactivation

When you backdate a cancellation or inactivation, the refunds for the events that occurred during the backdated period are prorated and applied accordingly.

The PCM_OP_SUBSCRIPTION_CYCLE_FORWARD opcode determines the scale values that are used by real-time rating to prorate the cycle fee amount to be refunded. This opcode prorates and applies the cycle fee amount to be refunded when a product or discount is canceled or inactivated.

When you backdate the cancellation of a product with cycle forward events, cycle forward fees are prorated and refunded from the backdated cancellation date.

**Example of backdated product cancellation**

When you backdate the cancellation of a product with cycle forward events, cycle forward fees are prorated and refunded from the backdated cancellation date.

For example, consider that you create an account on September 1 with the DOM set to 1. On November 5, you backdate the cancellation of the product with the following details to September 15:

- Cycle forward fee $3.00.
- Proration set to **Charge based on amount used**.

The three cycles including the current cycle are refunded as follows:

- $1.50 for the period of September 15 to October 1 (for 15 days).
- $3.00 for the period of October 1 to November 1.
- $3.00 for the period of November 1 to December 1.

These refunds are included in the next bill that gets generated during the bill run on December 1.

**Example of backdated discount cancellation**

On April 1, consider that you create an account that owns a product with $50 as a cycle forward fee and a discount of 10% on the monthly cycle fees. A cycle fee of $50 and a discount of $5 is applied for the period of April 1 to May 1, making the balance $45.

On April 28, you cancel the discount backdated to April 16. The discount of $2.50 applied for the period of April 16 to May 1 is refunded, making the balance as $47.50.

### How Earned and Unearned Revenue Is Set for Backdated Period

In the case of cycle forward events, the charges for the backdated period have the EARNED_START_T (**billing_cycle_start**) and EARNED_END_T (**billing_cycle_ end**) set to the backdated period.

For example, on February 15, a product with cycle forward events for an account with the DOM set to 1 is purchased, backdated to January 15. The EARNED_ START_T and EARNED_END_T for the backdated period (January 15 through February 1) will be January 15 and February 1 respectively, while the EARNED_START_T and EARNED_ END_T for the current cycle will be February 1 and March 1 respectively.

### Backdating a Product, Discount, or Deal Cancellation

To backdate a product, discount, or deal cancellation, enter the backdated time in the PIN_FLD_ END_T field of the corresponding opcode. For example, enter the date to which you want to backdate the cancellation in the PIN_FLD_ END_T field of the PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT opcode to backdate a product cancellation.

## Getting Data about Deals, Products, Discounts, and Services

Use the following opcodes to get data about deals, products, discounts, and services:

- PCM_OP_CUST_POL_GET_PLANS. See "Getting Plans, Deals, and Products for Purchase".
- PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS. See "Getting a List of Deals, Products, Discounts, and Services".
- PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS. See "Getting a List of Plans and Deals That an Account Owns".
- PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS. See "Reading Data for All Valid Purchased Products and Discounts".
- PCM_OP_SUBSCRIPTION_GET_HISTORY. See "Finding Events Associated with Deals, Products, Discounts, and Services".

## Getting Plans, Deals, and Products for Purchase

For more information on purchasing plans, deals, and products, see "Managing Customers' Services and Products".

Use the following opcodes to get plans, deals, and products for customer purchase:

- To get a list of plans for customer purchase, use the PCM_OP_CUST_POL_GET_ PLANS policy opcode. This policy opcode chooses the plans to return based on the AAC fields in the input flist.

  If the account is new, this policy opcode gets plans from the *new* plan list. Otherwise, it uses the *addon* plan list. If it does not find a list, the *default* plan list is used.

  You can customize the PCM_OP_CUST_POL_GET_PLANS policy opcode to search the **/group/plan_list** object by customer type and display the correct plan lists based on the customer type.

For more information on plan lists, see "About Plan Lists" in *BRM Setting Up Pricing and Rating*.

- To get a list of deals for customer purchase, use the PCM_OP_CUST_POL_GET_DEALS policy opcode. This policy opcode searches for all **/deal** objects that are valid (deal **END_T = 0** is valid). Each deal in the list is then read and checked to see whether the deal's **permitteds** array allows the storable object type to purchase it.

  Use the PCM_OP_CUST_POL_READ_PLAN policy opcode to customize deals during account creation. For a given plan, this policy opcode constructs a tree of services, deals, and products associated with that plan. This policy opcode retrieves account-level plans in addition to plans related to services.

- To get a list of products for customer purchase, use the PCM_OP_CUST_POL_GET_PRODUCTS policy opcode. The product's **permitteds** array is checked for valid storable object types purchasing the product.

## Getting a List of Deals, Products, Discounts, and Services

To get the hierarchical relationships of deals, products, discounts, and services associated with an account, use the PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS opcode. This policy opcode is used by Customer Center when a request is made to view a list of an account's deals, products, discounts, and services in a hierarchical format.

By default, PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS does not return "item" or "canceled and deleted" products and discounts. Set PIN_FLD_FLAGS in the input flist to get item, canceled, and deleted product and discount instances. For these records, the information about the product and discount instances are retrieved from the **/purchased_product** and **/purchased_discount** purchase and cancellation event objects.

> **Note:** Item, canceled, and deleted product and discount instances are not returned if the product's or discount's event objects are not recorded or if they are purged. For example, if BRM is configured to not record events with no balance impacts, records for these events are not returned.

PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS gets the purchase, cycle, and usage validity periods for products and discounts from the account's **/purchased_product** and **/purchased_discount** objects. It returns this information in the purchase, cycle, and usage START_T and END_T fields located in the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays in the output flist. If a product or discount is set to start on first usage and its validity period has not yet been initialized, additional information about the purchase, cycle, and usage start and end times is returned in START_DETAILS and END_DETAILS fields.

## Getting a List of Plans and Deals That an Account Owns

To get a list of the plans that an account owns, use the PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS policy opcode. This policy opcode retrieves a list of plans, deals, or both that an account owns.

**Note:** The policy opcode returns plans or deals that an account *owns*. It does not return a plan if the plan has only optional deals. The plan must have at least one purchased deal.

The PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS policy opcode takes an account POID from the input flist and returns a list of all plans, including balance group information and deals the account owns.

If the account contains optional deals that it does not own, those deals are returned with the list as eligible to purchase (PIN_FLD_BOOLEAN value of **0**).

The PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS policy opcode can perform additional filtering of plans or deals before they are returned. For example, add filtering logic to this policy opcode to return only inactive optional deals.

Plans and deals that were closed during a transition are not added to the output flist.

**Caution:** An account owning more than one instance of a plan is not supported. In this case, this policy opcode returns deals for one of the plans selected at random.

The PCM_OP_CUST_POL_GET_SUBSCRIBED_PLANS policy opcode returns deals in a **PIN_FLD_DEALS** array, even if it found them listed in the account in the older **PIN_FLD_DEAL_OBJ** field.

## Reading Data for All Valid Purchased Products and Discounts

To get the purchased products and discounts for accounts and services, use the PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS opcode.

The value of the PIN_FLD_SCOPE_OBJ field in the input flist restricts the results to an account, bill unit (**/billinfo** object), or service. This field is required and can specify one of the following values:

- An **/account** object: The opcode gets all the products and discounts for the account and its services.
- A **/billinfo** object: The opcode gets the purchased products and discounts associated with the specified bill unit.
- A **/service** object: The opcode gets the purchased products and discounts that belong to the specified service.

The opcode performs a search based on the specified scope object. Most of the other parameters are filters that work on a particular scope. The input flist looks like this:

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 618010 0
0 PIN_FLD_SCOPE_OBJ       POID [0] 0.0.0.1 /service/ip 615706 3
0 PIN_FLD_STATUS_FLAGS    INT [0] 3
0 PIN_FLD_END_T           TSTAMP [0] (1154415600) Tue Aug  1 00:00:00 2006
0 PIN_FLD_VALIDITY_FLAGS  INT [0] 3
0 PIN_FLD_INCLUSION_FLAGS INT [0] 2
0 PIN_FLD_OVERRIDE_FLAGS  INT [0] 2
0 PIN_FLD_SELECT_RESULT   INT [0]
0 PIN_FLD_OVERRIDDEN_OBJ  POID [0] 0.0.0.1 /purchased_product 324706 0
0 PIN_FLD_DEAL_OBJ        POID [0] 0.0.0.1 /deal 615702 3
0 PIN_FLD_PACKAGE_ID      INT [0] 23
0 PIN_FLD_PRODUCTS        ARRAY [*]     NULL array ptr
0 PIN_FLD_DISCOUNTS       ARRAY [*]     NULL array ptr
```

Use the following parameters to limit the search further:

- To retrieve only products or only discounts, use the PIN_FLD_PRODUCTS and PIN_FLD_DISCOUNTS arrays.
- To retrieve a limited number of fields, specify those fields under the PRODUCTS and DISCOUNTS arrays.
- To retrieve products and discounts with a specific status, pass the PIN_FLD_STATUS_FLAGS field and specify one of the following values:
  - PIN_SUBS_FLG_OFFERING_STATUS_ACTIVE: Retrieves only active offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_INACTIVE: Retrieves only inactive offerings.
  - PIN_SUBS_FLG_OFFERING_STATUS_CLOSED: Retrieves only closed offerings.

  **Note:** Using multiple values implies the target object can satisfy any of them.

- To retrieve products or discounts valid as of a specified time, pass the PIN_FLD_END_T field in the input flist. You can base the end time on only the purchase, cycle, or usage period by including the PIN_FLD_VALIDITY_FLAGS field in the input flist with one or more of the following values:

  - PIN_SUBS_FLG_OFFERING_VALIDITY_CYCLE: The opcode compares the specified END_T value with the CYCLE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_PURCHASE: The opcode compares the specified END_T value with the PURCHASE_END_T value.
  - PIN_SUBS_FLG_OFFERING_VALIDITY_USAGE: The opcode compares the specified END_T value with USAGE_END_T value.

  **Note:** Using multiple flags implies the target object must satisfy all of them.

- To fetch *all* eligible products and discounts whose validity period has the latest end time, set the input flist's PIN_FLD_SELECT_RESULT field to **4.** If that field is set to another value or omitted, the opcode returns only the first product or discount with the latest end time that it finds.

  **Note:** This opcode fetches product and discount records from the audit table in the BRM database. In that table, records are sorted on the end date of their validity period. Within a particular date group, however, the records are sorted randomly. Therefore, the opcode returns them in a random order. To control the order in which such records are displayed in an invoice, you must customize the invoicing feature to sort the records on an appropriate invoice field.

- To retrieve all eligible account and subscription offerings, pass the PIN_FLD_INCLUSION_FLAGS field on the input flist with one or more of the following values:
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_PRODS: Retrieves all eligible products.
  - PIN_SUBS_FLG_OFFERING_INCLUDE_ALL_ELIGIBLE_DISCS: Retrieves all eligible discounts.

  **Note:** When this field is omitted, only eligible offerings from a specified scope are returned.

- To retrieve only account products and discounts, pass the PIN_FLD_OVERRIDE_FLAGS field set to PIN_SUBS_FLG_OFFERING_ACCT_LEVEL_ONLY. This flag is valid for **/account** objects only.
- To retrieve customized products that override the base product, pass the following two fields in the input flist:
  - PIN_FLD_OVERRIDE_FLAGS field set to PIN_SUBS_FLG_OFFERING_OVERRIDE_PRODS_ONLY. When a valid product POID is passed in the OVERRIDDEN_OBJ field, this flag returns all the customized products that override the specified base product. When a NULL product POID is sent, only the base products are returned. This field is valid for any scope.
  - PIN_FLD_OVERRIDDEN_OBJ set to the POID of the base product that is overridden. This flag searches for all the products that use that base product, including the base product itself. If the OVERRIDDEN_OBJ field is null and

the PIN_FLD_OVERRIDE_FLAGS value is set to PIN_SUBS_FLG_OFFERING_OVERRIDE_PRODS_ONLY, only base products are retrieved.

- To retrieve products and discounts for only a single plan, pass the PIN_FLD_PACKAGE_ID field set to the package ID. This field can be used with any scope.
- To retrieve products and discounts for a specific deal, pass the PIN_FLD_DEAL_OBJ field set to the **/deal** object's POID.

PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS returns all purchased products and discounts, including canceled products and discounts, within the specified parameters.

## Finding Events Associated with Deals, Products, Discounts, and Services

Use the PCM_OP_SUBSCRIPTION_GET_HISTORY opcode to retrieve the event history for a deal, product, discount, or service instance associated with an account.

This opcode retrieves events for a deal, product, discount, or service instance in a specified date range. Which events are returned depends on the category specified in the input flist:

- If a *deal* is specified, a history of all events associated with the specific deal instance is returned for the account. The deal is identified by the PIN_FLD_PACKAGE_ID field in the input flist.
- If a *product* is specified, a history of all events associated with the specific product instance is returned for the account. The product instance is identified by the PIN_FLD_OFFERING_OBJ field in the input flist.
- If a *discount* is specified, a history of all events associated with the specific discount instance is returned for the account. The discount instance is identified by the PIN_FLD_OFFERING_OBJ field in the input flist.
- If a *service* is specified, all events associated with that service object are returned for the account.

If you do not specify a date range, PCM_OP_SUBSCRIPTION_GET_HISTORY uses the account's creation time.

If a RESULTS array is passed in the input flist, PCM_OP_SUBSCRIPTION_GET_HISTORY returns only fields from the events that are passed in the RESULTS array.

## Retrieving Product Details

Use the PCM_OP_PRICE_GET_PRODUCT_INFO opcode to gather product information, including:

- Pipeline rate plan data, if the product includes events configured for pipeline rating.
- Provisioning tag details from the **/config/provisioning_tag** object or the **/config/telco/*** object, if a product is configured with a provisioning tag. See "Working with Provisioning Tags" in *BRM Setting Up Pricing and Rating*.

The opcode searches for the **/product** object you specify in the input flist. You can specify either the POID of the object or a type-only POID and the product name.

The opcode examines each PIN_FLD_USAGE_MAP array in the product information returned by the search. Based on the content of the array, the opcode searches for and retrieves several different types of rating information:

- The opcode retrieves the content of all real-time rate plans included in the usage map, as well as the content of the **/rate** objects included in those rate plans.
- If a usage map array includes an event to be rated by a pipeline rate plan, the opcode optionally retrieves pipeline rate plan data from the database. You specify whether the opcode retrieves pipeline rate plan data by passing the FM_PRICE_SUPPRESS_PIPELINE_DATA flag in the opcode call:
  - When the FM_PRICE_SUPPRESS_PIPELINE_DATA flag is set (**0x0001**) in the opcode call, the opcode retrieves the pipeline rate plan data from the database.
  - When the FM_PRICE_SUPPRESS_PIPELINE_DATA flag is not set (**0**) in the opcode call, the opcode does not retrieve any pipeline rate plan data.

  For more information on passing flags in opcode calls, see "Understanding the PCM API and the PIN Library" in *BRM Developer's Guide*.
- If it includes a **/rate_plan_selector** object, the opcode retrieves its contents.
- If it includes a **/rollover** object, the opcode retrieves its contents.

The opcode returns the product details in the output flist.

If the **/product** object includes a provisioning tag attribute, the opcode performs the following:

- For **/service/telco/*** service types, the opcode retrieves the provisioning tag details from the **/config/telco/*** object and returns it in the output flist's PIN_FLD_PROVISIONING_TAG_INFO array.
- For all other service types, the opcode retrieves the provisioning tag details from the **/config/provisioning_tag** object and returns it in the output flist's PIN_FLD_PROVISIONING_TAG_INFO array.

If the product is customized, the PIN_FLD_TAILORMADE field has the value **1**. The PIN_FLD_TAILORMADE_DATA field includes information about customized real-time rates, stored as semicolon-delimited, comma-separated pairs of resources and percentages.

## Creating Customized /Product Objects

Use the PCM_OP_PRICE_PREP_TAILORMADE_PRODUCT opcode to assemble the data required to create or modify a customized **/product** object.

The input to the opcode includes the following:

- The POID of the base product to be customized. A type-only POID can be used instead if the product name is supplied.
- The product's customized real-time rate plans and rates.
- The product's customized pipeline pricing data, including rate plan versions, rate plan configurations, price models, model selectors, and model selector rule sets.
- A list of customized pipeline pricing customizations, expressed in string format as semicolon-delimited, comma-separated pairs of resource IDs and percentages. For example, 20% discounts on US dollar and minutes resources are represented as **USD,-20;MIN,-20**.

During the creation of a customized product, the opcode does the following:

- Retrieves full product and pipeline rate plan details.
- Applies the percentage increase or decrease specified in the customization to the applicable real-time rate's balance impacts.

- Creates a name for the customized product by prepending the current time in seconds (expressed in hexadecimal) to the base product name. See "About Customized Product and Pipeline Pricing Component Names".
- For each pipeline rate plan configuration in the input flist, creates copies of price models and model selectors and applies percentage increases or decreases when necessary.
- Renames customized pipeline pricing components. The code of each component is changed to include a two-letter abbreviation, the current time in milliseconds (expressed in hexadecimal), and a unique integer value. See "About Customized Product and Pipeline Pricing Component Names".
- Returns an flist for the complete customized **/product** object, including both original and customized data for both real-time and pipeline rate plans.

You pass the output of this opcode to the PCM_OP_PRICE_SET_PRICE _LIST opcode, which commits the customized **/product** object to the BRM database.

When the opcode is called to modify an existing customized product, the PIN_FLD_POID and PIN_FLD_NAME fields in the PIN_FLD_PRODUCTS array contain the actual values for the existing customized **/product** object.

The PIN_FLD_TAILORMADE_DATA fields contain the new customization percentages and resources. The opcode updates the current real-time and pipeline rates based on the new percentages. If the changes affect pipeline rate plans that were not previously customized, the opcode creates copies of pricing data that is now required.

## Validation Rules for Products, Discounts, Deals, and Plans

- A product cannot be canceled if it belongs to a required deal or to a deal that has a dependency relationship.
- A deal cannot be transitioned if the source service and the target service are not the same service.
- A deal cannot be transitioned if the associated service is inactive.
- A deal cannot be transitioned if it is required in the associated plan.
- A plan cannot be transitioned if the source service and the target service are not the same service or if the service is inactive.

## Creating Custom Transition Types for Deals and Plans

By default, BRM provides these transition types for plans and deals:

- Upgrade
- Downgrade
- Generation Change

To add custom transition types for deals and plans:

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 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 on the utility's syntax, see "load_transition_type".

5. Restart Pricing Center.

Your new transition types are loaded into the **/config/transition_type** object and are displayed in Pricing Center the next time you start it.

## Transition Type API Considerations

The default transition types and your custom transition types are stored in the **/config/transition_type** object in the PIN_FLD_TRANSITIONS array; for example:

```
0 PIN_FLD_TRANSITIONS       ARRAY [1]
1   PIN_FLD_TYPE             ENUM [0] 1
1   PIN_FLD_TYPE_STR          STR [0] "Upgrade"
0 PIN_FLD_TRANSITIONS       ARRAY [2]
1   PIN_FLD_TYPE             ENUM [0] 2
1   PIN_FLD_TYPE_STR          STR [0] "Downgrade"
0 PIN_FLD_TRANSITIONS       ARRAY [3]
1   PIN_FLD_TYPE             ENUM [0] 3
1   PIN_FLD_TYPE_STR          STR [0] "Generation Change"
0 PIN_FLD_TRANSITIONS       ARRAY [101]
1   PIN_FLD_TYPE             ENUM [0] 101
1   PIN_FLD_TYPE_STR          STR [0] "RED"
```

The example shows these transition types:

- Upgrade - number 1
- Downgrade - number 2
- Generation Change - number 3
- RED - number 101

**Note:** The element ID of the array is the same as its PIN_FLD_TYPE.

You could customize a policy opcode to read this object for validation checks.

# 12

# Changing Account and Service Status

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

It also describes how to manage deferred actions.

For more information on changing service states, see "About Triggering State Changes in Custom Service Life Cycles".

## About Activating, Inactivating, and Closing Accounts

An account status can be active, inactive, or closed.

You can backdate account status changes to a date earlier than the current date. You can schedule the status change for a future date. You change account status on the **Change Account/Service Status** panel in Customer Center.

For more information on the procedures for changing account and service status, see Working with products, deals, discounts, and services.

- When an account is *active*, the customer can use all active services.
- When an account is *inactive*, the customer cannot use any service. Typically, the customer can still use Web administration forms when the account is inactive. For example, if a payment cannot be made because of a credit card verification error, the customer can use the Web administration form to update credit card and address information so the payment can be made.

  **Note:** Inactivating an account prevents customers from generating usage and cycle balance impacts but does not prevent the account from being charged for bills already due. See "Billing Inactive Accounts".

- When an account is *closed*, the customer cannot use any service. A closed account is normally closed permanently, but you can reactivate it in Customer Center if it was closed by mistake. Closing an account does not delete the account or its service login names and passwords from the BRM database. Because closed accounts are never deleted from the database, there is no time limit on when they can be reactivated.

  **Important:** Closing an account cancels all the products owned by the account.

> **Note:** You cannot delete closed accounts from a production database, but you can reuse their login names. See "Reusing Login Names and Passwords from Closed Accounts or Canceled Services".

You can customize what a customer can do when an account is closed or inactive. For example, you can allow customers to view their account status on the Web.

The only functional difference between a closed account and an inactive account is that you can enable BRM to activate and inactivate accounts automatically. BRM cannot automatically close an account or automatically activate a closed account.

## Account and Service Status

Changing the status of an account also changes the status of all the account's services. Changing the status of a *service* affects only that service.

- When you inactivate an account, all the account's services are inactivated.
- When you reactivate an inactive account, all the account's services that were inactivated when the account was inactivated are reactivated. Services that were inactivated independently of the account status are not reactivated.
- When you close an account, all the account's services are closed.
- When you reactivate a closed account, all the account's services that were closed when the account was closed are reactivated. Services that were manually inactivated (whose status was not changed when the account was closed) remain in the inactive state when the account is re-activated.

> **Note:** When you close an account, all the products owned by the account are canceled. When you re-activate the account, you need to re-purchase the products.

See "Activating and Inactivating Services".

## Product and Discount Status

Changing the status of an account also changes the status of all the account's active products and discounts.

- When you inactivate an account, all the account's products and discounts are inactivated.

> **Note:** In Customer Center, the inactivated products and discounts continue to be displayed on the account's **Product** tab.

- When you reactivate an inactive account, you reactivate all products and discounts that were active when the account was active. Products and discounts that were inactive when the account was active remain inactive.
- When you close an account, all the account's products and discounts are canceled.

> **Note:** In Customer Center, the canceled products and discounts are removed from the account's **Product** tab.

- When you reactivate a closed account, the account's products and discounts are *not* reactivated. To regain the canceled products and discounts, the account must repurchase the deals that contain them.

Reactivating a closed account does not reinstate the canceled products and discounts. To regain the canceled products and discounts, the account must repurchase the deals that contain them.

## About Backdated Status Change

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

BRM does not allow backdating the status change if:

- 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, product or discount. 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, consider that you change the status of an account from active to inactive on September 1. On October 1, you cannot backdate a status change 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.

### Backdating Status Changes

To backdate a status change, enter the date to which you want to backdate the account, service, product, or discount status change in the PIN_FLD_END_T field of the applicable opcodes. For example, enter the backdated date in the PIN_FLD_ END_T field of the following opcodes:

- PCM_OP_CUST_SET_STATUS: for backdating the account status change.
- PCM_OP_CUST_UPDATE_SERVICES: for backdating the service status change.
- PCM_OP_CUST_SET_PRODUCT_STATUS: for backdating the product status change.
- PCM_OP_CUST_SET_DISCOUNT_STATUS: for backdating the discount status change.

## About Status Changes and Balance Impacts

If a product status changes from active to inactive or closed before the end of the billing cycle:

- If the product includes a cycle forward rate that allows proration, the unused fee is refunded.
- If the product includes a cycle arrears rate that allows proration, the used amount is charged.

## Billing Inactive Accounts

When you inactivate an account, you also inactivate all its services, which prevents the customer from using the services. BRM does not charge usage and cycle fees while the account is inactive. But while the account is inactive, BRM continues to collect bills that were due before the account was inactivated.

## About Rating and Service Status

In most cases, if a service is inactive or closed, the service is unprovisioned on the network, and no events can be generated by using that service. However, there might be occasions where a call is made even though the service has been inactivated or closed in BRM but is still provisioned on the network. In that case, BRM still rates the event.

## About Plan Transitions and Service Status

If a service was closed because it was associated with a plan transition, its status flag value is set to PIN_STATUS_FLAG_DUE_TO_TRANSITION, and you cannot change the closed status of the service. For more information on transition rules, see "Transitioning Deals".

## Setting Permissions for Changing Account Status

You can set Customer Center permissions to restrict who can change the status of accounts. See "Setting up Permissions in BRM Applications" in *BRM System Administrator's Guide*.

## Scheduling Status Changes in Advance

In Customer Center, you can schedule account and service status changes for a future date. You can use a daily billing utility, **pin_deferred_act**, to change the status automatically on the scheduled date. See "Executing Deferred Actions With the pin_deferred_act Utility" in *BRM Configuring and Running Billing*.

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.

Customer Center lists all deferred actions scheduled for an account. From the list, you can cancel an action or execute it immediately.

For more information, see "Managing Deferred Actions".

## Automatically Inactivating and Closing Accounts

BRM can automatically inactivate and reactivate accounts even when no deferred action is scheduled. 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, 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_SPEC_VERIFY

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

## Inactivating and Closing Accounts in Hierarchical Groups

By default, changing the status of a parent account changes the status of child accounts and bill units in the parent's hierarchical group in the following ways:

- The status of all the parent account's bill units are changed.
- The status of every subordinate bill unit in every child account is changed.

> **Note:** The child account's status and the status of nonsubordinate bill units in child accounts are *not* changed.

- If a subordinate bill unit in a child account is also the parent of another account's bill units, the status of the subordinate bill units in the other account is also changed.

For more information, see "How Bill Unit Status Changes Affect Hierarchies" in *BRM Managing Accounts Receivable*.

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.
- In a sponsor group, if the sponsor group owner account is inactive, the member accounts receive the balance impacts that normally would have been sponsored.

## Reusing Login Names and Passwords from Closed Accounts or Canceled Services

By default, you can not 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.

## Changing Purchase, Usage and Cycle Start Time for Reactivated Products and Discounts

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

This setting does not affect products whose purchase is deferred to a later date. For those products, if the product is reactivated before the purchase date, the original purchase, usage, and the cycle start times are always 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 product and discount.

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 you installed the BRM software.
2. Add the following entry:

   ```
   -fm_bill change_start_time_on_activation 0
   ```

   - **0** keeps the original purchase, usage, and cycle start times.
   - **1** changes the purchase start time, usage start time, and cycle start time to the reactivation date. This is the default.
3. Save and close the file.
4. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## Allowing Active Services with Inactive Accounts

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

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 1
   ```

   where

   - **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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## How BRM Changes Product Status

To set the status of a **/purchased_product** object owned by an account, use the PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS opcode.

PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS is called in the following cases:

- When the status of an account or service is changed.
- When a product status is changed. You might need to change only the product status itself if, for example, the product was purchased as inactive because of future provisioning and you activate it later.

> **Note:** When you change the status of a base product, the opcode automatically changes the status of any associated customized products. You cannot set the status of a customized product directly.

PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS performs the following tasks:

1. Opens a transaction.
2. Retrieves the new status from the PIN_FLD_STATUSES array in the input flist. If the product status change is due to an account or service status change, PIN_STATUS_FLAG_DUE_TO_ACCOUNT is passed in the PIN_FLD_STATUS_FLAGS field of the array.
3. Reads the old status of the product from the database.
4. If PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS is *not* called in CALC_ONLY mode, applies the new status to the product.
5. If the product is inactive due to future provisioning and is now being activated, then the PCM_OP_SUBSCRIPTION_SET_PRODINFO opcode is called to modify the purchase start date and time to the date and time of reactivation. If the original cycle and usage start date is earlier than the new purchase start date, PCM_OP_SUBSCRIPTION_SET_PRODINFO also resets the cycle and usage start date and time to the new purchase start date. For more information, see "Handling Purchase, Cycle, and Usage Start and End Times".

If PCM_OP_SUBSCRIPTION_SET_PRODINFO is not called, an **/au_purchased_products** audit object is generated, which is used for rerating events.

6. If PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS is called while activating an account, sets the purchase time to NOW and sets the usage and cycle fees by calling PCM_OP_SUBSCRIPTION_SET_PRODINFO.
7. If the product status change is backdated, validates that:
   - The date to which the product status change is backdated is not prior to the G/L posting date.
   - The date to which the product status change is backdated is not prior to the account or service's effective date.
   - The date to which the product status is backdated is not prior to the last status change of the account or service. See "About Backdated Status Change".
8. Creates an audit log event object (**/event/billing/product/action/modify/status**).
9. Closes the transaction.

If the PCM_OP_CALC_ONLY flag is set when calling PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS, it returns the entire event flist for the events created as a result of the modification. Otherwise, it returns the event Portal object IDs (POIDs) of all event objects created as a result of the modification.

## How BRM Changes Discount Status

For general information about modifying a discount in a deal, see "Modifying Discount Attributes".

To set the status of a **/purchased_discount** object owned by an account, use the PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS opcode.

This opcode is called when the status of a discount is changed. This can occur in the following cases:

- When the status of the account or service that owns the discount is changed. In this case, the discount's status is changed to the status of the account or service.
- When the status of a discount is changed from inactive to active.

PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS performs the following tasks:

1. Validates and sets the new status. The new status cannot be the same as the old status.

   **Note:** When you change the status of a base product, the opcode automatically changes the status of any associated customized products. You cannot set the status of a customized product directly.

2. Calls the PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO opcode to reset the **/discount** object's purchase, cycle, and usage start and end dates.

   **Note:** If the discount purchase, cycle, and usage start or end dates are already set to a later date, or the CM is configured to not reset dates, the dates are not reset.

3. If the status change of the discount is backdated, this opcode validates that:

- The date to which the discount status change is backdated is not prior to the general ledger (G/L) posting date.
- The date to which the discount status change is backdated is not prior to the account or service effective date.
- The date to which the discount status is backdated is not prior to the last status change of the account or service. See "About Backdated Status Change".

4. Generates an **/event/billing/discount/action/modify/status** event to record the status change.

If successful, PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS returns the POID of the **/event/billing/discount/action/modify/status** event.

# Setting Account, Service, and Bill Unit Status by Using Your Custom Application

For more information on changing status, see "Changing Account and Service Status".

The following opcodes are used to set account, bill unit, and service status:

- PCM_OP_CUST_SET_STATUS. See "Changing the Status of an Account, Bill Unit, or Service".

> **Important:** Do not call this opcode directly for service status changes. Call the PCM_OP_CUST_UPDATE_SERVICES opcode instead.

- PCM_OP_CUST_POL_PREP_STATUS and PCM_OP_CUST_POL_VALID_STATUS. See "Customizing Status Changes".
- PCM_OP_ACT_POL_EVENT_LIMIT. See "Inactivating Accounts that Exceed a Specified Limit".

## Changing the Status of an Account, Bill Unit, or Service

The PCM_OP_CUST_SET_STATUS opcode changes and checks the status of an account, bill unit, or service. You can call the opcode directly for accounts and bill units. For service status changes, use the PCM_OP_CUST_UPDATE_SERVICES opcode, which in turn calls PCM_OP_CUST_SET_STATUS. See "Modifying Services" for more information.

In addition to changing status, PCM_OP_CUST_SET_STATUS does the following:

- Triggers auto-billing if bills are still pending. See "About Auto-Triggered Billing" in *BRM Configuring and Running Billing*.
- Calls opcodes to prorate cycle fees.

To return all the fields in the event object, set the PCM_OPFLG_READ_RESULT flag. If this flag is not set, PCM_OP_CUST_SET_STATUS returns only the POID.

If PCM_OP_CUST_SET_STATUS is called at the account level, more than one result can be returned:

- One for each service owned by this account.
- One for each bill unit owned by this account.
- One or more for each child account and subordinate bill unit.

- One for each deferred action.

To run PCM_OP_CUST_SET_STATUS without changing data in the database, use the PCM_OPFLG_CALC_ONLY flag.

- If the flag is set, no fields in the database are changed and the event object is not created. The fields used to create the event object and adjustment item are returned to the caller.
- If the flag is *not* set, the **/event/customer/status** storable object is created to record details of the operation.

### PIN_FLD_STATUS Field Values

When you use PCM_OP_CUST_SET_STATUS, set the account or bill unit status entry to one of the following field values listed in Table 12–1. See *BRM_Home*/**include/pin_cust.h.**

***Table 12–1 PIN_FLD_STATUS Values***

| PIN_FLD_STATUS Field | Description | Value |
|---|---|---|
| PIN_STATUS_ACTIVE | Account or bill unit fully functional; login is subject to credit limit check. | **10100** |
| PIN_STATUS_INACTIVE | Inactive account or bill unit; no logins permitted and no fees are charged. | **10102** |
| PIN_STATUS_CLOSED | Closes the account or bill unit, which can be reopened later. This does not activate the corresponding products. | **10103** |
| PIN_STATUS_DEFUNCT | Reserved for BRM.<br>PIN_ERR_BAD_ARG is returned if you try to set a status to this value. | **0** |

### PIN_FLD_STATUS_FLAG Field Values

The values used to further define the status for the account, bill unit, or service are listed in Table 12–2:

***Table 12–2 PIN_FLD_STATUS_FLAG Values***

| PIN_FLD_STATUS_FLAG Field | Description | Value |
|---|---|---|
| PIN_STATUS_FLAG_ACTIVATE | Sets the future activation date (accounts and bill units only). | **0x01** |
| PIN_STATUS_FLAG_DEBT | Displays the outstanding debt (accounts and bill units only). | **0x02** |
| PIN_STATUS_FLAG_MANUAL | Displays manual operations performed by the administrative operator. | **0x04** |
| PIN_STATUS_FLAG_DUE_TO_ACCOUNT | Closes all related services for the account or bill unit (accounts and bill units only). | **0x08** |
| PIN_STATUS_FLAG_DUE_TO_PARENT | Closes all child accounts if the parent account is closed (only for accounts in groups set up for billing purposes).<br>Closes all child bill units if the parent bill unit is closed. | **0x10** |
| PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE | Changes all member services when the subscription's service status is changed. | **0x20000** |

*Table 12–2 (Cont.) PIN_FLD_STATUS_FLAG Values*

| PIN_FLD_STATUS_FLAG Field | Description | Value |
|---|---|---|
| PIN_STATUS_FLAG_PO_ EXHAUSTED | Obsolete. | **0x20** |
| PIN_STATUS_FLAG_ PROVISIONING | Identifies tasks that use the provisioning system. | **0x40** |

The field values in the storable object are set as follows:

- If the input status is PIN_STATUS_ACTIVE:
  - Flags = (Old flag AND NOT (New flag))
  - If the result flag is **0**, new status is input status, else old status.
- If the input status is PIN_STATUS_INACTIVE:
  - The **status** field is set to inactive.
  - The **flags** field is set to the **OR** of the old flags in the database and the new **flags** on the input flist.
- If the input status is PIN_STATUS_CLOSED:
  - The **status** field is set to **closed**.
  - The **flags** field is set to the **OR** of the old flags and new flags.

If the status change is caused by a change to the parent account or bill unit, PIN_FLD_STATUS_FLAG_DUE_TO_PARENT is set in the new flags.

If the status change is caused by a change to an accounts receivable (AR) account or AR bill unit, PIN_FLD_STATUS_FLAG_DUE_TO_ACCOUNT is set in the new flags.

Accounts, bill units, or services marked as **closed** (terminated) are actually closed when the **pin_deferred_act** billing application is run. See "Executing Deferred Actions With the pin_deferred_act Utility" in *BRM Configuring and Running Billing*.

### Deferred Actions

For deferred actions, the PIN_FLD_WHEN_T field is passed to indicate the date on which the status change will be made. It also creates a **/schedule** object to store the input flist to call PCM_OP_CUST_SET_STATUS on the specified date.

You can create, delete, execute, and modify **/schedule** objects using Customer Center or by calling the opcodes in Table 12–3:

*Table 12–3 Opcodes used to Modify /schedule Objects*

| Opcode | Function |
|---|---|
| PCM_OP_ACT_SCHEDULE_CREATE | Creates a schedule object. |
| PCM_OP_ACT_SCHEDULE_DELETE | Deletes a schedule object. |
| PCM_OP_ACT_SCHEDULE_MODIFY | Modifies a schedule object. |
| PCM_OP_ACT_SCHEDULE_EXECUTE | Executes a schedule object. |

By default, the **pin_deferred_act** utility, which is part of the **pin_bill_day** billing script, finds expired schedule objects and calls the PCM_OP_ACT_SCHEDULE_EXECUTE opcode, which in turn calls PCM_OP_CUST_SET_STATUS.

If the deferred status change is for closing an account, bill unit, or service, PCM_OP_CUST_SET_STATUS sets the PIN_FLD_CLOSE_WHEN_T field in the account, bill unit, or service to midnight on the specified date.

#### Setting, Resetting, or Unsetting a Deferred CLOSE

When setting or resetting a deferred CLOSE, PCM_OP_CUST_SET_STATUS calls a standard opcode to set the PIN_FLD_PURCHASE_END_T, PIN_FLD_CYCLE_END_T, and PIN_FLD_USAGE_END_T fields for each corresponding product and discount to the deferred CLOSE date if the old values for these fields are later than the deferred CLOSE date or if these fields are set to NEVER (“0”).

If a deferred CLOSE is canceled (the schedule object is deleted), PCM_OP_CUST_SET_STATUS calls a standard opcode to set the values of the PIN_FLD_PURCHASE_END_T, PIN_FLD_CYCLE_END_T, and PIN_FLD_USAGE_END_T fields for all corresponding products and discounts to the old values from the previous **/event/billing/product/action/modify** event.

### Customizing Status Changes

You can customize status changes by using the following opcodes:

- To customize the way status is changed, use the PCM_OP_CUST_POL_PREP_STATUS policy opcode. The default implementation does nothing.
- To validate status changes, use the PCM_OP_CUST_POL_VALID_STATUS policy opcode. The default is to do no additional checking and to return the verified information.

  For example, you can use this policy opcode to customize how customer service representatives (CSR) can set permissions on specific service types.

See "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

### Inactivating Accounts that Exceed a Specified Limit

Use the PCM_OP_ACT_POL_EVENT_LIMIT policy opcode to inactivate any account or account hierarchy that exceeds a specified limit.

PCM_OP_ACT_POL_EVENT_LIMIT performs the following tasks:

1. Calls PCM_OP_CUST_SET_STATUS to inactivate an account or account hierarchy based on the status set in the PIN_FLD_STATUS field.
2. Calls the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode to notify any custom applications that a limit was reached and that an action took place.

## Managing Deferred Actions

In Customer Center, 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 hierarchical group.
- Removing an account from a hierarchical group.
- Moving an account between hierarchical groups.

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

To execute deferred actions automatically on their scheduled date, you can use a daily billing utility, **pin_deferred_act**. See "Executing Deferred Actions With the pin_deferred_act Utility" in *BRM Configuring and Running Billing*.

### Displaying Deferred Actions in Customer Center

In Customer Center, the *total* number of deferred actions scheduled for an account is shown on the **Summary** tab in the **Account Summary** section. This number includes the following actions:

- All actions whose execution date is in the future. The status of these actions is **Pending**.
- Any actions that could not be executed on their scheduled date and have not been deleted from the Deferral Details table. The status of these actions is **Error**.
- Any completed actions that have not been deleted from the Deferral Details table. The status of these actions is **Done**.

The number of deferred actions scheduled for *each service* owned by an account is shown in the **Deferred Actions** column of the table on the **Service** tab.

## Managing Deferred Actions in Your Custom Application

Use the following opcodes to handle deferred actions:

- PCM_OP_ACT_SCHEDULE_CREATE. See "Scheduling Deferred Actions".
- PCM_OP_ACT_SCHEDULE_MODIFY. See "Modifying Deferred Action Descriptions".
- PCM_OP_ACT_SCHEDULE_DELETE. See "Deleting Deferred Actions".
- PCM_OP_ACT_SCHEDULE_EXECUTE. See "Executing Deferred Actions".
- PCM_OP_ACT_POL_VALIDATE_SCHEDULE. See "Performing Policy Checks before Scheduling Deferred Actions".

### Scheduling Deferred Actions

The PCM_OP_ACT_SCHEDULE_CREATE opcode creates **/schedule** objects, which schedule a single action for a predetermined date and time.

BRM supports the following deferred actions:

- Account activation, deactivation, and closure.
- Account parent change.
- Service activation and closure.

Two optional fields, PIN_FLD_WHEN_T and PIN_FLD_DESCR, enable deferred actions in BRM. These fields are set by the input flists of the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode or the PCM_OP_BILL_GROUP_MOVE_MEMBER opcode.

- PIN_FLD_DESCR describes the deferred action.
- PIN_FLD_WHEN_T specifies when the deferred action occurs.

If the PIN_FLD_WHEN_T field is populated in the input flist of the calling opcode, that opcode can call PCM_OP_ACT_SCHEDULE_CREATE to create a **/schedule** object to hold the input flist information. The **/schedule** object remains active until the PIN_

FLD_WHEN_T time expires and the PCM_OP_ACT_SCHEDULE_EXECUTE opcode executes the action stored in the **/schedule** object.

> **Note:** Deferred actions are always rounded off to midnight. For example, if you use PCM_OP_ACT_SCHEDULE_CREATE to schedule a deferred action on January 15 at 8:00 a.m., the schedule object is created with a scheduled time of January 15 at 00:00.

## Modifying Deferred Action Descriptions

The PCM_OP_ACT_SCHEDULE_MODIFY opcode modifies an existing **/schedule** object.

This opcode modifies the text description in PIN_FLD_DESCR field and the scheduled date in PIN_FLD_WHEN_T field.

## Deleting Deferred Actions

PCM_OP_ACT_SCHEDULE_DELETE deletes existing **/schedule** objects.

## Executing Deferred Actions

PCM_OP_ACT_SCHEDULE_EXECUTE executes any deferred actions defined in a **/schedule** object. This opcode is called directly by the **pin_deferred_act** billing utility.

When you run the **pin_deferred_act** utility, it searches for any **/schedule** object with both an expired PIN_FLD_WHEN_T field and a PIN_FLD_STATUS field marked **Pending**. When it finds one, it calls PCM_OP_ACT_SCHEDULE_EXECUTE to:

1. Execute the action defined in the **/schedule** object.
2. Update PIN_FLD_STATUS to either **Done** or **Error**, depending on its success.

## Performing Policy Checks before Scheduling Deferred Actions

Use the PCM_OP_ACT_POL_VALIDATE_SCHEDULE policy opcode to perform custom policy checks before creating a **/schedule** object. For example, you can customize this opcode to verify that an account balance is zero before scheduling it for closure.

# 13

# Managing Service Life Cycles

This chapter describes how to use the Service Lifecycle Management (SLM) feature to configure and use custom service life cycles in Oracle Communications Billing and Revenue Management (BRM).

## About Service Life Cycles

The life cycle of a service is composed of the states and state transitions through which the service passes from its initial state to its final state.

By default, all BRM service types use the default service life cycle, which has the following statuses:

- **Active:** The customer can perform normal service activity.
- **Inactive:** The customer's use of the service is restricted, usually because a credit limit was exceeded, a bill was not paid, or the service is new and is waiting for provisioning. This is a temporary status.
- **Closed:** The customer is prevented from using any aspect of the service. This status implies that the customer will not use the service again.

You cannot customize the default service life cycle. 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. BRM includes a sample prepaid service life cycle that has the following states:

- **Preactive:** This is the start state of the sample prepaid service life cycle. The subscriber has not yet used the service.
- **Active:** The subscriber can perform normal service activity.
- **Recharge Only:** The subscriber can receive calls and use free aspects of the service, but she cannot make prepaid calls. She can use top-ups to replenish the balance.
- **Credit Expired:** The subscriber cannot make or receive calls.
- **Dormant:** 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. The subscriber cannot make or receive calls.
- **Suspended:** The subscriber intends to resume the service. In this state, no aspect of the service can be used.
- **Closed:** This is the final state of the prepaid service life cycle. In this state, no aspect of the service can be used.

See "About the Sample Prepaid Service Life Cycle" for more information.

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

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

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

**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.

## About Using Custom Service Life Cycles

To use custom service life cycles, do the following:

1. Set up BRM to support custom service life cycles:
   a. Enable the SLM feature. See "Enabling BRM to Use Custom Service Life Cycles".
   b. Enable the **load_config** utility to validate SLM configuration files. See "Enabling load_config to Validate SLM Configuration Files".
   c. Add SLM entries to the Connection Manager (CM) **pin.conf** file. See "Adding SLM Entries to the CM pin.conf File".
2. Create a custom service life cycle. See "About Creating Custom Service Life Cycles".
3. Map each state in the custom service life cycle to a status in the default service life cycle. See "About Mapping States to Statuses".
4. Associate one or more services with the custom service life cycle. See "About Associating Services with Custom Life Cycles".
5. Associate account bill units with the SLM business profile. See "Associating Bill Units with the SLM Business Profile".
6. Enable Services Framework AAA Manager to validate AAA requests for services that use the custom life cycle. See "Validating AAA Requests for Services that Use Custom Life Cycles".
7. Configure Customer Center to display the names of the custom service life cycle states. See "Configuring Customer Center to Display Custom Service Life Cycle States".

## Setting Up BRM to Support Custom Service Life Cycles

To set up BRM to support custom service life cycles, do the following:

- Enable the SLM feature. See "Enabling BRM to Use Custom Service Life Cycles".

- Add the SLM validation entry to the **load_config pin.conf** file. See "Enabling load_config to Validate SLM Configuration Files".
- Add entries for custom service life cycles to the CM **pin.conf** file. See "Adding SLM Entries to the CM pin.conf File".

## Enabling BRM to Use Custom Service Life Cycles

By default, BRM supports only the default service life cycle.

Before you can create and use custom service life cycles, you must configure BRM to support them by enabling the customer **SubscriberLifeCycle** business parameter in the **business_params** object.

To enable BRM to use custom service life cycles:

1. Go to the *BRM_Home*/**sys/data/config** directory, where *BRM_Home* is the directory in which you installed the BRM software.
2. Create an editable XML file for the **customer** parameter class by using the following command:

   ```
   pin_bus_params -r  BusParamsCustomer bus_params_customer.xml
   ```

   This command creates the XML file named **bus_params_customer.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.
3. Open the **bus_params_customer.xml.out** file
4. Search for the following line:

   ```
   <SubscriberLifeCycle>disabled</SubscriberLifeCycle>
   ```

5. Change **disabled** to **enabled**.
6. Save the file as **bus_params_customer.xml**.
7. Run the following command, which loads this change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_customer.xml
   ```

   **Caution:** BRM uses the XML in this file to overwrite the existing **customer** 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 **customer** configuration.

   **Note:** To run this command from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
8. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

   For more information on 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## Enabling load_config to Validate SLM Configuration Files

You use the **load_config** utility to load the following service life cycle information:

- Custom service life cycles configured in the **config_lifecycle_states.xml** file. See "About Creating Custom Service Life Cycles" for more information.
- Mapping between custom service life cycle states and the default service life cycle statuses configured in the **config_service_state_map.xml** file. See "About Mapping States to Statuses" for more information.

To validate those XML files, **load_config** needs the following entry in its **pin.conf** file:

```
- load_config validation_module libLoadValidSLM LoadValidSLM_init
```

If that entry is not in the utility's **pin.conf** file, add it before using the utility to load the XML files.

> **Note:** The **pin.conf** file is in the directory from which the utility is run. The utility is located in *BRM_Home*/**apps/load_config**. See "load_config" for more information.

## 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 **pin.conf** file in *BRM_Home*/**sys/cm**.
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, that you plan to create in your system. See "About 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, that you plan to create in your system. See "About 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 "About 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 "About 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*.

For example, if the only custom life cycle you use is the sample prepaid service life cycle and the only business profile you use is the default SLM business profile and its three service validation templates, Oracle recommends the following settings:

```
- cm_cache fm_bill_utils_business_profile_cache 1, 12040, 1
- cm_cache fm_bill_template_cache 3, 10240, 1
- cm_cache fm_cust_lifecycle_config_cache 1, 102400, 1
- cm_cache fm_cust_statemap_config_cache 1, 102400, 1
```

3. Save and close the file.
4. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

# About Creating Custom Service Life Cycles

To create custom service life cycles, edit the service life cycle configuration file (*BRM_Home*/**sys/data/config/config_lifecycle_states.xml**). See "About the Service Life Cycle Configuration File" for more information.

By default, that 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.

After editing the configuration file, use the **load_config** utility to load the file's contents into a **/config/lifecycle_states** object in the BRM database. See "Creating Custom Service Life Cycles" for more information.

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

## About 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 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 (see "About the Service Life Cycle Configuration File" for more information).

Customer Center does not allow CSRs to make unauthorized state changes. See the Customer Center Help for more information.

- **The first use of a service:** Services that use the sample prepaid service life cycle start in the Preactive state. The first use of such services triggers the PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE helper opcode to change the state from Preactive to Active.
- **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.

  For example, this policy opcode supports the sample prepaid service life cycle as follows:

  - If the service state is Active and a balance impact causes the resources associated with the service to reach their credit limit, this policy opcode calls the PCM_OP_CUST_UPDATE_SERVICES opcode, which calls the PCM_OP_CUST_SET_STATUS opcode to change the state to Recharge Only. The state expiration time in the **/service** object is not updated.
  - If the service state is Recharge Only or Credit Expired and a balance impact replenishes the associated resources, this policy opcode calls the same opcodes to change the state to Active.
  - If the service state is Preactive, Active, or Credit Expired and a voucher is used to replenish the balance, this policy opcode compares the voucher's validity period with the state's expiration period. It then updates the state expiration date in the **/service** object based on the greater of the two periods.

  **Note:** Oracle recommends that top-ups not be performed in the Preactive state.

  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** (see "About the Service Life Cycle Configuration File"). If this flag is not set to **1** in any of the transitions configured for a state, this utility does not change the state.

  For example, **pin_state_change** supports the sample prepaid service life cycle as follows:

- When a service is in the Active state and it reaches its state expiration date, the utility changes the state to Recharge Only.
- When a service is in the Recharge Only state and its balance is not replenished by the state's expiration date, the utility changes the state to Credit Expired.
- When a service is in the Credit Expired state and its balance is not replenished by the state's expiration date, the utility changes the state to Suspended.

See "pin_state_change" for more information.

See "About Triggering Sample Prepaid State Changes" for more information.

See "Changing Account and Service Status" for more information on how service status changes are triggered.

## About 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. 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.

You configure business rules in the **<RULES>** child element of the **<STATES>** element in the **config_lifecycle_states.xml** file (see "About the Service Life Cycle Configuration File" for more information). 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.)

  For example, the business rules in the sample prepaid service life cycle are used by the AAA module. In that module, the PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE and PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE opcodes evaluate each request against the business rules configured for the service's current state and authorize or reject the request based on those rules.
- **<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. See "Creating Custom Service Life Cycles" for more information.

See "About the Sample Business Rules" for more information on the business rules in the sample prepaid service life cycle.

# 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.
2. Add a **<ConfigObject>** element to the **<ObjectList>** element.

3. In the **<ConfigObject>** element, specify values for the elements listed in Table 13–1, " Custom Service Life Cycle Elements".
4. Save and close the file.
5. Run the following command, which loads the **config_lifecycle_states.xml** file:

   ```
   load_config config_lifecycle_states.xml
   ```

   **Important:**

   - 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_lifecycle_
     states.xml
     ```

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

   For more information on 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## About the Service Life Cycle Configuration File

You configure all the custom service life cycles in your BRM system in the *BRM_Home*/**sys/data/config/config_lifecycle_states.xml** file. By default, this file contains a sample prepaid service life cycle (see "About the Sample Prepaid Service Life Cycle").

Each life cycle is defined in a **<ConfigObject>** element, which has the following syntax:

```
<ConfigObject>
   <DESCR>Description</DESCR>
   <NAME>Name</NAME>
   <PERMITTED_SERVICE_TYPES elem="0">
      <SERVICE_TYPE>ServiceType</SERVICE_TYPE>
   </PERMITTED_SERVICE_TYPES>
   <STATES elem="0">
      <INITIAL_STATE>InitialStateInteger</INITIAL_STATE>
      <SERVICE_STATE_EXPIRATION_T>days:hrs:mins</SERVICE_STATE_
         EXPIRATION_T>
      <STATE_ID>StateID</STATE_ID>
      <STATE_NAME>StateName</STATE_NAME>
```

```
      <RULES elem="0">
         <MODULE>Module</MODULE>
         <NAME>Name</NAME>
         <VALUE>Value</VALUE>
      </RULES>
      <TRANSITIONS elem="0">
         <DEFAULT_FLAG>DefaultFlag</DEFAULT_FLAG>
         <NEXT_STATE>NextState</NEXT_STATE>
         <POST_OPCODE>PostOpcode</POST_OPCODE>
         <PRE_OPCODE>PreOpcode</PRE_OPCODE>
      </TRANSITIONS>
   </STATES>
</ConfigObject>
```

Table 13–1 lists the child elements of the **<ConfigObject>** element:

***Table 13–1 Custom Service Life Cycle Elements***

| XML Element | Description | Notes |
|---|---|---|
| **DESCR** | (Optional) Character string that describes the service life cycle. | Minimum length is 0 characters.<br>Maximum length is 255 characters. |
| **NAME** | Character string used as the name of the service life cycle.<br>The value of this element is used in the key-value pair that associates a service with the custom service life cycle. See "About Associating Services with Custom Life Cycles". | The name must be unique within your BRM system.<br>Minimum length is 1 character.<br>Maximum length is 255 characters. |
| **PERMITTED_ SERVICE_TYPES** | Parent element of a **<SERVICE_TYPE>** child element.<br>This element is mapped to the PIN_FLD_ PERMITTED_SERVICE_TYPES array in the **/config/lifecycle_states** object. The array lists the service types to which the states in this life cycle apply. | If you configure multiple **<PERMITTED_ SERVICE_TYPES>** 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.<br>**Note:** The array is included for your convenience. It is not used in the validation process, and it does not associate services with this life cycle. See "About Associating Services with Custom Life Cycles" for more information. |
| **SERVICE_TYPE** | Character string that specifies a service type in the PIN_FLD_PERMITTED_SERVICE_ TYPES array. | Specify any service type in your BRM system, such as **/service/telco/gsm/telephony**.<br>Minimum length is 1 character.<br>Maximum length is 255 characters. |
| **STATES** | Parent element of the child elements that define a state for this life cycle.<br>This element is mapped to the PIN_FLD_ STATES array in the **/config/lifecycle_states** object. The array contains all the states configured for this life cycle. | 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. |
| **INITIAL_STATE** | Integer that specifies whether this is the initial state of the life cycle. | Specify one of the following values:<br>▪ **1:** Initial state. Only one state in the life cycle can have this value.<br>▪ **0:** Not the initial state. |

***Table 13–1 (Cont.) Custom Service Life Cycle Elements***

| 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.<br>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:<br>■ *days*<br>■ *days*:*hrs*<br>■ *days*:*hrs*:*mins*<br>Where<br>■ *days* is the number of days.<br>■ *hrs* is the number of hours.<br>■ *mins* is the number of minutes.<br>For example, 365:0:600 specifies 365 days, 0 hours, and 600 minutes.<br>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.<br>Each state ID must be unique.<br>The sample prepaid service life cycle uses the following state IDs:<br>■ **101:** Preactive<br>■ **102:** Active<br>■ **103:** Recharge Only<br>■ **104:** Credit Expired<br>■ **105:** Dormant<br>■ **106:** Fraud Investigated<br>■ **107:** Suspended<br>■ **108:** Closed<br>See "About the Sample Prepaid Service Life Cycle" for more information on these states. |
| **STATE_NAME** | Character string used as the name of the state (for example, **Active**).<br>This string is mapped to the PIN_FLD_STATE_NAME field in the **/config/lifecycle_states** object. | Minimum length is 1 character.<br>Maximum length is 255 characters.<br>**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. |

*Table 13–1 (Cont.) Custom Service Life Cycle Elements*

| XML Element | Description | Notes |
|---|---|---|
| **RULES** | Parent element of the child elements that define a business rule for this state.<br>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 life cycle state.<br>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.<br>See "About 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.<br>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.<br>Minimum length is 1 character.<br>Maximum length is 255 characters. |
| **NAME** | Character string used as the name of the business rule (for example, **REQ_ALLOWED**).<br>**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.<br>Minimum length is 1 character.<br>Maximum length is 255 characters. |
| **VALUE** | Character string that specifies whether the rule is enabled or disabled. | Specify one of the following values:<br>■ **Yes:** Enabled<br>■ **No:** Disabled |
| **TRANSITIONS** | Parent element of the child elements that define how this state changes to another state.<br>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.<br>For more information on how transitions are triggered, see "About 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 indicates whether this is the default transition for this state.<br>When the **pin_state_change** utility changes a service state, it uses the PIN_FLD_NEXT_STATE whose PIN_FLD_DEFAULT_FLAG is set to **1** in the PIN_FLD_TRANSITIONS array. | Specify one of the following values:<br>■ **1:** Default transition. Only one transition for the state can have this value.<br>■ **0:** Not the default transition. |

*Table 13–1 (Cont.) Custom Service Life Cycle Elements*

| XML Element | Description | Notes |
|---|---|---|
| **NEXT_STATE** | Integer that identifies the state to change the service to. | Specify the state's unique ID (see "STATE_ID"). |
| **POST_OPCODE** | Integer that identifies the policy opcode you want BRM to call after the state transition occurs.<br>In this policy opcode, you must code the action to perform immediately after a state transition occurs. For example, you might add logic to the policy opcode that notifies the subscriber to let her know that her 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 *BRM_Home***/include** directory.<br>Enter **0** to call no opcode. |
| **PRE_OPCODE** | Integer that identifies the policy opcode you want BRM to call before the state transition occurs.<br>In this policy opcode, you must code the action to perform immediately before a state transition occurs. For example, you might add logic to the policy opcode that notifies the subscriber to let her know that her 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 *BRM_Home***/include** directory.<br>Enter **0** to call no opcode. |

## 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.

   **Important:** 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".

   See Table 13–1, " Custom Service Life Cycle Elements" for more information on the elements of a custom service life cycle.

3. Set the **configMode** attribute of the **<ObjectList>** element to one of the following values:

   - **replace:** (Default) Updates the existing **/config/lifecycle_states** objects.
   - **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
   ```

**Important:**

- 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_lifecycle_
  states.xml
  ```

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

   For more information on 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### Deleting Custom Service Life Cycles

To delete *all* custom service life cycles from your BRM system, run the following command:

```
load_config -r lifecycle_states
```

**WARNING: Do not delete a life cycle that a service is using. Doing so will cause data corruption.**

This command deletes all instances of the **/config/lifecycle_states** class from your BRM database.

See "load_config" in *BRM Developer's Guide* for more information.

## About Mapping States to Statuses

Each service life cycle state (for example, Preactive, Active, Recharge Only, Credit Expired, Fraud Investigated, Dormant, Suspended, and Closed in the sample prepaid service life cycle) 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 subscription 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.

By default, that file contains mapping for the sample prepaid service life cycle, which you can use as is, modify, or delete. See "About the Sample Service State-to-Status Mapping" 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.

The states shown in Table 13–2 are configured as the status defaults for the sample prepaid service life cycle:

*Table 13–2 Default States for Statuses in the Sample Prepaid Service Life Cycle*

| Status | Default State |
|---|---|
| Active | Active |
| Inactive | Suspended |
| Closed | Closed |

**Important:** 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.

The sample prepaid service life cycle has two states that are exceptions to this rule: Preactive (the start state) and Closed (the end state).

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

## Mapping States to Statuses

**Important:** 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 13–3, " Service State Mapping Elements".
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
   ```

**Important:**

- 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 more information on 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## 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. By default, this file contains mapping for the sample prepaid service life cycle (see "About the Sample Prepaid Service Life Cycle").

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 13–3 lists the elements of the preceding syntax:

*Table 13–3 Service State Mapping Elements*

| XML Element | Description | Notes |
|---|---|---|
| **DESCR** | (Optional) Character string that describes the state-to-status mapping. | Minimum length is 0 characters.<br>Maximum length is 255 characters. |
| **NAME** | Character string used as the name of the mapping. | The name must be unique within your BRM system.<br>Minimum length is 1 character.<br>Maximum length is 255 characters. |
| **STATES** | Parent element of the child elements that map a state to a status.<br>This element is mapped to the PIN_FLD_STATES array in the **/config/service_state_map** object. The array must contain all the the service life cycle states in your system. It can contain states from multiple service life cycles.<br>See "About 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.<br>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:<br>■ **1:** Default state. If multiple states are mapped to a status, only one of those states can have this value.<br>■ **0:** Not the default state. |

*Table 13–3 (Cont.) Service State Mapping Elements*

| XML Element | Description | Notes |
|---|---|---|
| **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.<br>This ID is configured in the **config_lifecycle_states.xml** file (see Table 13–1, " Custom Service Life Cycle Elements").<br>The sample prepaid service life cycle uses the following state IDs:<br>■ **101:** Preactive<br>■ **102:** Active<br>■ **103:** Recharge Only<br>■ **104:** Credit Expired<br>■ **105:** Dormant<br>■ **106:** Fraud Investigated<br>■ **107:** Suspended<br>■ **108:** Closed<br>See "About the Sample Prepaid Service Life Cycle" for more information on these states. |
| **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:<br>■ **10100:** Active<br>■ **10102:** Inactive<br>■ **10103:** Closed |
| **STATUS_FLAGS** | Integer that specifies the status flag to pass when a state change occurs.<br>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.<br>**Note:** STATUS_FLAGS values are listed in the *BRM_Home*/**include/pin_cust.h** file. |

## 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 13–3, " Service State Mapping Elements" for more information on 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
   ```

**Important:**

- 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 more information On 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### 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
```

**WARNING: 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.

## About 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, you 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 (see Table 13–1, " Custom Service Life Cycle Elements")

The SLM business profile is configured in the **pin_slm_business_profile.xml** file. By default, it associates the **/service/telco/gsm/telephony** service type with the sample prepaid service life cycle (see "About the Sample Prepaid Service Life Cycle").

**WARNING: 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.**

**For example, if subscribers begin purchasing the /service/telco/gsm/telephony service when it uses the default life cycle (Active, Inactive, Closed), do not later associate it with the sample prepaid life cycle.**

See "Associating Services with Custom Life Cycles" for more information on editing the SLM business profile and loading it into the BRM database.

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

## Associating Services with Custom Life Cycles

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 on **<TemplateID>** elements.

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

   By default, these key values are associated with this business profile:

   ```
   <NameValue key="Prepaid" value="yes" />
   <NameValue key="CacheResidency" value="REALTIME" />
   ```

   See "Defining Business Profiles" for more information on **<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.

   By default, these validation templates are defined in the SLM business profile:

   ```
   <TemplateList>
      <Template name="ServiceTelcoGprs" type="/service/telco/gprs">
         <Desc>Description of the template</Desc>
         <Iscript />
         <NameValue key="Prepaid" value="yes" />
         <NameValue key="CacheResidency" value="REALTIME" />
      </Template>
      <Template name="ServiceTelcoGsm" type="/service/telco/gsm">
   ```

```
      <Desc>Description of the template</Desc>
      <Iscript />
      <NameValue key="Prepaid" value="yes" />
      <NameValue key="CacheResidency" value="REALTIME" />
   </Template>
   <Template name="ServiceTelcoGsmTel" type="/service/telco/gsm/
      telephony">
      <Desc>Description of the template</Desc>
      <Iscript />
      <NameValue key="Prepaid" value="yes" />
      <NameValue key="CacheResidency" value="REALTIME" />
      <NameValue key="lifecycle_obj" value="Default Lifecycle Configuration" />
   </Template>
</TemplateList>
```

**Note:** In the **ServiceTelcoGsmTel** template definition, the following key-value pair associates the **/service/telco/gsm/telephony** service with the sample prepaid service life cycle:

- **key = lifecycle_obj**
- **value = Default Lifecycle Configuration** (the NAME value of the sample prepaid service life cycle in the default **config_lifecycle_states.xml** file)

See "About Associating Services with Custom Life Cycles".

See "Defining Validation Templates" for more information on the **<TemplateList>** and **<Template>** elements.

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**

   See the following for more information:

   - "Creating Service and Event Storable Classes" in *BRM Developer's Guide*
   - "About Validation Templates"
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
   ```

**Important:**

- 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 more information on 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

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 (**/billinfo** object) must be associated with the business profile in which the service is linked to the custom life cycle (see "About Associating Services with Custom Life Cycles").

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. See "Making the SLM Business Profile Your System's Default Business Profile" for more information.
- 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 "Overriding the Default Business Profile" in *BRM System Administrator's Guide* for more information.

## Making the SLM Business Profile Your System's Default Business Profile

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

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

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

2. Search the XML file for following line:

   ```
   <DefaultBusinessProfile>string</DefaultBusinessProfile>
   ```

3. Change *string* to **SLM** (the name of the business profile configured in (**pin_slm_business_profile.xml**):

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

   **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, the changes affect the associated aspects of BRM's billing configuration.

4. Save and close the file.
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. Go to the *BRM_Home*/**sys/data/config** directory.
7. Run the following command, which loads the changes into the **/config/business_params** object in the BRM database:

   ```
   pin_bus_params bus_params_billing.xml
   ```

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

   For more information on 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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
10. Make sure the PIN_FLD_BILLINFO array is correctly populated when the account is created. See discussions about the following for more information:
    - PCM_OP_CUST_CREATE_ACCT opcode
    - PCM_OP_CUST_COMMIT_CUSTOMER opcode
    - PCM_OP_CUST_CREATE_CUSTOMER opcode

- Creating **/billinfo** objects in *BRM Configuring and Running Billing*

See "Changing the Default Business Profile" in *BRM System Administrator's Guide* for more information.

# Validating AAA Requests for Services that Use Custom Life Cycles

When Services Framework AAA Manager performs AAA for a service that uses a custom life cycle, it must call a helper opcode during the VALIDATE_LIFECYCLE processing stage to validate the service request against the business rules configured for the current state of the service. By default, it calls the PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE helper opcode.

To enable Services Framework to do that, you must map any service that uses a custom life cycle to the appropriate AAA opcode, processing stage, and helper opcode in the AAA opcode map (**/config/opcodemap/tcf** object).

See "Configuring Services Framework to Call Helper Opcodes" in *BRM Telco Integration* for more information on creating AAA opcode mapping.

See *BRM Telco Integration* for more information on AAA processing stages.

## Adding VALIDATE_LIFECYCLE Entries for the Sample Prepaid Service Life Cycle

To support the sample prepaid service life cycle, the following entries must be in the AAA opcode map:

```
Framework_Opcode: PCM_OP_TCF_AUTHORIZE
Processing_Stage: VALIDATE_LIFECYCLE
Opcode_Map: /service/telco/gsm/telephony, PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE

Framework_Opcode: PCM_OP_TCF_AAA_ACCOUNTING
Processing_Stage: VALIDATE_LIFECYCLE
Opcode_Map: /service/telco/gsm/telephony, PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE

Framework_Opcode: PCM_OP_TCF_AAA_STOP_ACCOUNTING
Processing_Stage: VALIDATE_LIFECYCLE
Opcode_Map: /service/telco/gsm/telephony, PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE
```

If those entries are not in the AAA opcode map, you must add them to it.

> **Note:** To check the entries in the AAA opcode map, display the **/config/opcodemap/tcf** object by using Object Browser or the **testnap** utility's **robj** command. For more information on 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*.

To add VALIDATE_LIFECYCLE entries for the sample prepaid service life cycle to the AAA opcode map:

1. Open the *BRM_Home***/sys/data/config/pin_config_opcodemap_tcf** file in a text editor.
2. Add the entries to the file.

> **Note:** If the entries are in the file but not in the AAA opcode map, the object was not loaded into the database after the entries were added to it.

3. Save and close the file.
4. Run the following command, which loads the changes into the **/config/opcodemap/tcf** object in the BRM database:

   > **Note:** To replace the entire contents of the object, use the **-f** parameter. To append data to the object, use the **-i** option.

   ```
   load_aaa_config_opcodemap_tcf -i|-f pin_config_opcodemap_tcf
   ```

   See "load_aaa_config_opcodemap_tcf" in *BRM Telco Integration* for more information.
5. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
6. Read the updated object with the **testnap** utility's **robj** command or with Object Browser to verify that all fields are correct.

   For more information on 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*.

## Configuring Customer Center to Display Custom Service Life Cycle States

To display the names of custom service life cycle states in Customer Center fields, lists, and tables, add the names to the *CC_Home***/lib/CustomizedResources.properties** file, where *CC_Home* is the directory in which you installed Customer Center. If this file does not exist, you must create it.

> **Note:** The default names of the sample prepaid service life cycle states are in the **CustomerCenterResources.properties** file. You do not have to add them to the **CustomizedResources.properties** file to see them in Customer Center.

To configure Customer Center to display the names of custom service life cycle states:

1. If your system does not have a **CustomizedResources.properties** file in the *CC_Home***/lib** directory, do the following:
   a. Copy the *CCSDK_Home***/CustomerCareSDK/CustCntr/custom/CustomizedResources.properties** file (where *CCSDK_Home* is the directory in which you installed the Customer Care SDK).
   b. Paste the copy into the *CC_Home***/lib** directory.
2. Add the following entry to the *CC_Home***/lib/CustomizedResources.properties** file:

   ```
   service.state.format={0,choice,0#No
   change|101#Preactive|102#Active|103#Recharge Only|104#Credit
   Expired|105#Dormant|106#Fraud Investigated|107#Suspended|108#Closed|state_
   ```

```
ID#state_name| . . . }
```

where

- *state_ID* is an integer that identifies a state in a custom service life cycle (for example, **101**). See "STATE_ID".
- *state_name* is a character string used as the name of the state (for example, **Preactive**). See "STATE_NAME".

The entry must include a *state_ID*#*state_name* element for every custom life cycle state in your system, *including* the states in the sample prepaid service life cycle.

If you want to customize the default prepaid service life cycle names, you can do so in this entry.

---

**Important:** By default, the **CustomerCenterResources.properties** file has a **service.state.format** entry that contains the names of the states in the sample prepaid service life cycle. The **service.state.format** entry in the **CustomizedResources.properties** file, however, overrides the entry in the **CustomerCenterResources.properties** file. Therefore, if you add the **service.state.format** entry to the **CustomizedResources.properties** file, make sure it includes the prepaid life cycle states.

---

For example, suppose that you want to add two states (Telephony1 ID 109 and Telephony2 ID 110) to the prepaid service life cycle and that you created another life cycle for SMS services that has three states (SMS1 ID 111, SMS2 ID 112, SMS3 ID 113). In that case, you would add the following entry:

```
service.state.format={0,choice,0#No
change|101#Preactive|102#Active|103#Recharge Only|104#Credit
Expired|105#Dormant|106#Fraud
Investigated|107#Suspended|108#Closed|109#Telephony1|110#Telephony2|111#SMS1|11
2#SMS2|113#SMS3}
```

3. Save and close the file.
4. Restart Customer Center.

# About the Sample Prepaid Service Life Cycle

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

- **Preactive:** This is the start 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:** This state 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:** This state indicates that 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:** This state 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.
- **Closed:** This is 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 13–4 lists the triggers that initiate state changes in the sample prepaid service life cycle:

***Table 13–4 Triggers for State Changes in the Sample Prepaid Service Life Cycle***

| Trigger | Change Performed By | From State | To State |
|---|---|---|---|
| First use of service | PCM_OP_TCF_AAA_ VALIDATE_LIFECYCLE | Preactive | Active |
| 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<br>or<br>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 | 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 |
| First use of service after specified period of inactivity | PCM_OP_TCF_AAA_ VALIDATE_LIFECYCLE | Dormant | Active |

For an overview of state-change triggers, see "About 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. This rule is evaluated by the PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE opcode. 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 is evaluated by the PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE policy opcode, which can also apply custom rules that you configure.
- **MT_ENABLED:** Specifies whether mobile-terminated calls are allowed. Values are **Yes** and **No**. This rule is evaluated by the PCM_OP_TCF_AAA_POL_VALIDATE_LIFECYCLE policy opcode, which 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 called CALL_ALLOWED, whose values represent various combinations of the values of all three of the preceding rules (see Table 13–5).

***Table 13–5 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 "About 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 mapping shown in Table 13–6. This mapping supports the states in the sample prepaid service life cycle.

***Table 13–6 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 |

*Table 13–6 (Cont.) State-to-Status Mapping for the Sample Prepaid Service Life Cycle*

| <LIFECYCLE_STATE> | <DEFAULT_FLAG> | <STATUS> | <STATUS_FLAGS> |
|---|---|---|---|
| **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 |

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

# 14

# Managing Customers' Subscription-Level Services

This chapter describes Oracle Communications Billing and Revenue Management (BRM) subscription service management, including setting up services to track and rate usage and create bills for customers' subscriptions.

Before reading this document, you should be familiar with managing customers. See "About Managing Customers".

## About Grouping Services by Subscription

BRM allows you to group services by subscription. A *subscription* is a service or group of services, such as telephony, email, and data transfer, that customers purchase for a particular device, such as a wireless phone. Customers might have several subscriptions, such as a wireless phone and Internet access or two wireless phones with different phone numbers.

Grouping services by subscription allows you to rate and bill for services according to the customer's subscription rather than by account. When services are grouped by subscription, you can:

- Keep separate balances for the subscription.
- Create a separate bill for the subscription.
- Offer subscription-level discounts.
- Specify subscription-level credit limits.

Subscription service management can be applied to various types of customers; for example:

- A single customer owns one wireless handset.
- A family in which one account owns multiple handsets.
- A corporate customer in which a sponsored account and subscription is set up for each employee.

You use Pricing Center to configure your subscription groups and Customer Center to purchase the subscription services and member services.

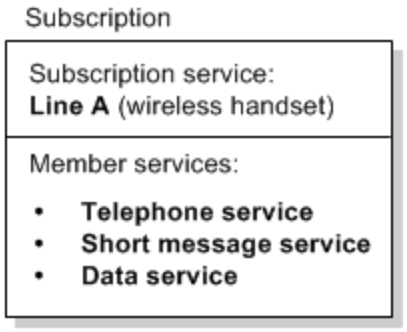
## About Subscription Services

To group services by subscription, you configure a subscription group in Pricing Center, which includes a subscription service, and assign member services:

- A *subscription service* represents a customer's subscription, such as a wireless phone line, and includes member services. You can associate the subscription service with the customer's device, such as the SIM card in a wireless handset.
- *Member services* are related to the subscription service. To purchase member services in Customer Center, you must also purchase its subscription service. You assign the devices associated with the subscription service to the member services when registering customers. For more information, see "Associating a Device with a Subscription Service".

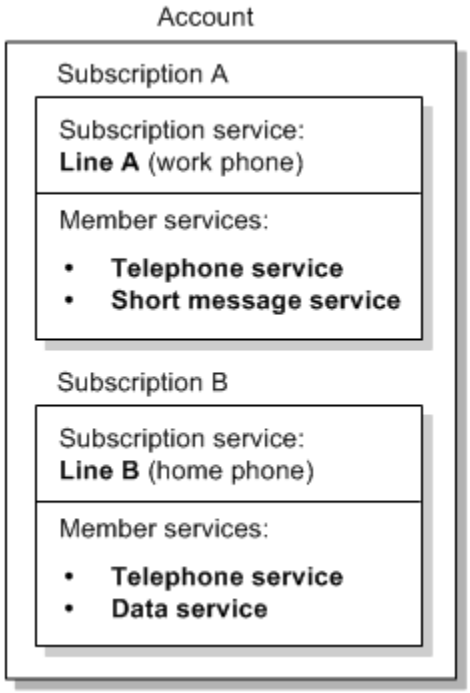
For example, a subscription might consist of the following services as shown in Figure 14–1:

***Figure 14–1 Services for a Subscription (Example)***




For example, a customer owns two wireless handsets, each with a unique phone number. You set up two subscription groups in the customer's account, one for each phone line as shown in Figure 14–2:

***Figure 14–2 Multiple Subscriptions for an Account***



## About Subscription Service Dependencies

A service instance cannot be a member of two different subscription services.

A subscription service cannot be a member of another subscription service.

However, a subscription service, a member service, or both can sponsor charges or share discounts with services in another subscription's group. For more information, see "About Sharing Discounts and Sponsoring Charges for Subscription Services".

## How Resources Are Tracked and Shared for Subscription Services

When a customer uses the services included in the subscription, the usage fees are stored in a balance group. You create balance groups to track balances for one or more specific services. A service in a subscription group can also include resources that can be distributed among one or more member services. How usage fees are tracked and resources are distributed is determined partly by the balance groups that you create for the services.

For more information on balance groups, see "About Accounts Receivable" in *BRM Configuring and Running Billing*.

### Tracking a Subscription's Currency Resources

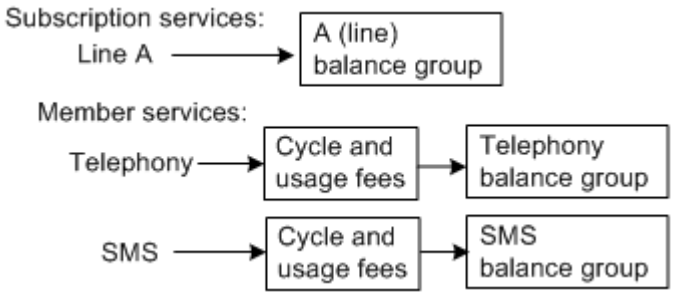
When you define the services for a subscription, you create one or more balance groups. To track resources at the subscription level, you define a balance group for the subscription service and associate all member services with that balance group.

While it is possible for a subscription service to use the account's balance group, it is much easier to track usage for a subscription when the subscription service has its own balance group. This is relevant when a customer's account includes both a subscription service, such as a handset with multiple services, and a standalone service such as an Internet connection.

> **Note:** You cannot transfer a subscription service that uses the account's balance group to another subscriber's account. If you want to be able to transfer the subscription service, define service-level balance groups or define a balance group for the subscription service and associate all the member services with that balance group. See "About Transferring a Subscription Service to Another Subscriber".

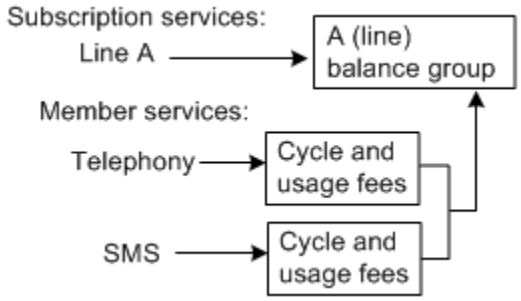
Typically, you also create a balance group for each member service. This makes it possible to set up shared discounts, set credit limits on service resources, and limit resource consumption to the specified service. When services have their own balance groups, cycle and usage fees for each service are tracked in the service's balance group as shown in Figure 14–3:

*Figure 14–3 Services and Balance Groups*




You can associate all services with the subscription service's balance group. When customers use member services, the cycle and usage fees are stored in the subscription service's balance group as shown in Figure 14–4:

*Figure 14–4 Storage of Cycle and Usage Fee*



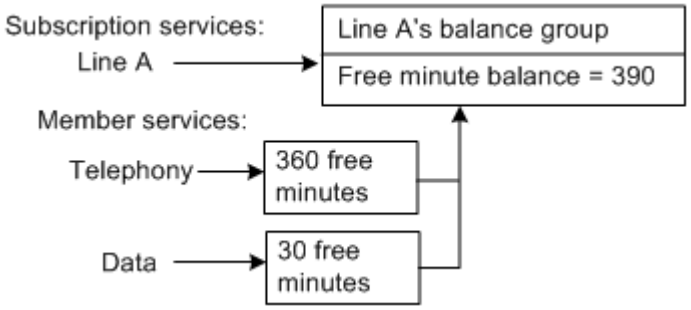

A member service can also be associated with another member service's balance group to track usage for those services together. However, member services cannot use the balance group of a service that is not in the same subscription group.

## Sharing a Subscription's Non-Currency Resources

When all member services use the subscription service's balance group, all non-currency resources, such as free minutes and frequent flyer miles, are shared by all services in the group.

For example, a telephony service includes 360 free minutes of usage per month and a data service includes 30 free minutes per month. At the beginning of the month, a balance of 390 free minutes is stored in the subscription service's balance group, and all services have access to the 390 free minutes as shown in Figure 14–5:

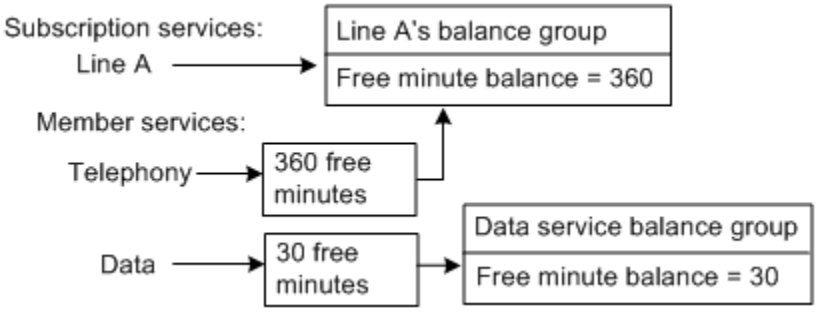
*Figure 14–5 Sharing of Non-Currency Resources*



When you create a balance group for a member service, any non-currency resources included with that service can be consumed by only that service.

In Figure 14–6, the data service includes 30 free minutes that can be applied only to the data service. The data service has access only to its own resources; it cannot use the free minutes included with any other service.

***Figure 14–6 Non-Currency Services with Multiple Balance Groups***




You might want some types of non-currency resources to be tracked 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 these non-currency resources at the subscription level, you configure a counting sub-balance for the subscription service. For more information, see "About Tracking Resources in Account Sub-Balances" in *BRM Setting Up Pricing and Rating*.

## How Products and Discounts Are Selected for Rating Subscription Service Usage

Products and discounts can be owned by the subscription service, a member service, or both:

- When the subscription service owns products and discounts, those products and discounts are used to rate member service usage.
- When the subscription service and member service both own a product, the product used for rating (the member's or the subscription service's) is determined by the product's priority.
- When the subscription service and member service both own discounts, the order in which the discounts are distributed is determined by the discount's priority.

If a member service owns a discount but does not have its own balance group, its discount adjustment is applied to the subscription service's balance group.

You specify product and discount priority when you create products and discounts in Pricing Center. For more information, see the Pricing Center Help.

## About Distributing Discounts in a Subscription Service Group

You can apply the same types of discounts to a subscription service as you can to a standalone service; for example:

- Cycle discounts, such as a 10% discount for certain months.
- Billing-time discounts, such as 10% off when the total usage fee exceeds $100.
- System discounts, such as 15% off on certain holidays for all customers.

- Shared discounts, such as free minutes that are distributed to services based on the service's usage amount.

For more information on discounts, see "About Implementing Discounts" in *BRM Configuring Pipeline Rating and Discounting*.

There are two ways that discounts owned by a subscription service can be distributed to member services in a subscription group:

- Through inheritance: When a member service shares the subscription service's balance group, any discount purchased by the subscription service is automatically inherited by the member service. The discount balance impact is applied to the subscription service's shared balance group.
- 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. For more information, see "About Sharing Discounts and Sponsoring Charges for Subscription Services".

**Note:** When a billing-time discount is inherited, it is applied to the subscription service and every member service that inherits it. If the discount grants a percentage off of currency charges 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, purchase them at the member service level instead of the subscription service level. You can also configure BRM to disallow inheritance for billing-time discounts. For more information, see "Specifying Whether Billing-Time Discounts Are Inherited by Member Services in Subscription Groups" in *BRM Configuring Pipeline Rating and Discounting*.

## About Sharing Discounts and Sponsoring Charges for Subscription Services

To apply subscription-level cycle, billing-time, and system discounts, you create the discounts and assign them to the subscription service. To apply shared discounts, you create a discount sharing group.

Charges can also be sponsored among a subscription's services. To sponsor charges, you create a charge sharing group.

Discount and charge sharing groups are groups of services that share resources. These resource sharing groups consist of group owners and group members. An owner can be any one of the following, and the members can include any combination of these:

- An account
- A subscription service
- A subscription's member service
- A service outside the subscription's group in the same or another account

To create subscription-level resource sharing (for example, to distribute 300 free minutes to all services on a phone line), make the subscription service the owner of the sharing group and make the member services the members of the sharing group.

The service type of the subscription service must match the type of service to which the shared discount applies.

**Important:** All services in a resource sharing group must have their own balance groups so that BRM can track the resources that they use and share.

For more information on discount and charge sharing groups, see "About Resource Sharing Groups" in *BRM Configuring Pipeline Rating and Discounting*.

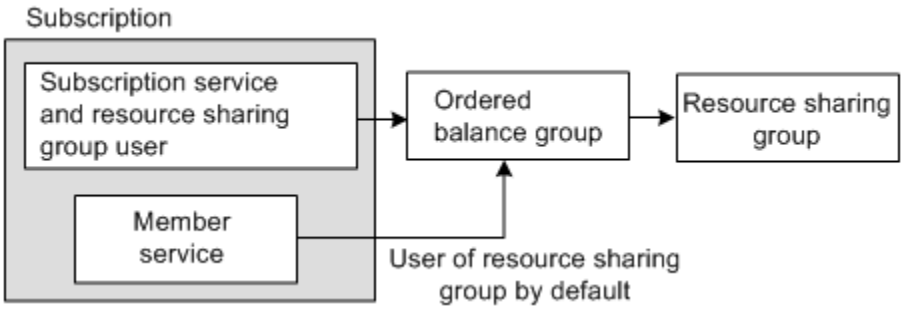
### How Sharing Group Resources Are Distributed to Subscription Service Members

Members in discount and charge sharing groups have an *ordered balance group*. For more information, see "About Ordered Balance Groups" in *BRM Configuring Pipeline Rating and Discounting*.

When a subscription service is a *member* in a resource sharing group, the member services may or may not share the subscription service's ordered balance group:

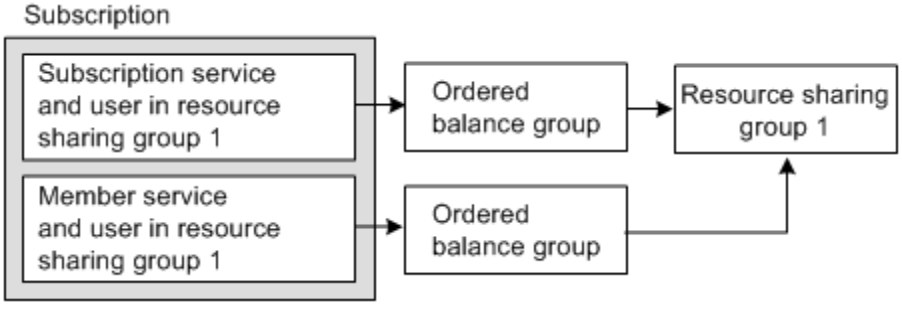
- A member service that is not itself a resource sharing group member (that is, it does not have its own ordered balance group), uses the ordered balance group of the subscription service. By default, the member service inherits member status in the resource sharing group from the subscription service, and the member service has access to the sharing group's resources as illustrated in Figure 14–7:

***Figure 14–7 Member Service Using Group's Resources***



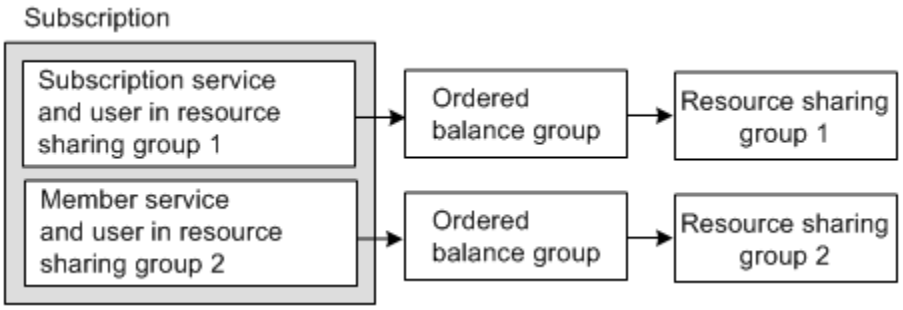

- A member service that has its own ordered balance group does not use the subscription service's ordered balance group. This is true even when the member service and subscription service belong to the same resource sharing group as illustrated in Figure 14–8:

***Figure 14–8 Member Service Using Own Ordered Balance Group***




- If a member service and the subscription service belong to different resource sharing groups, the services have access only to the shared resources in the sharing group to which they belong as illustrated in Figure 14–9:

*Figure 14–9 Restrictions for Services Using Different Resource 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 discount and charge sharing group, the discounts are used before charges are applied.

You set up discounts in Pricing Center.

You define how charges are shared in Pricing Center.

To create discount and charge sharing groups, you implement BRM opcodes in your customer relationship management (CRM) system.

## How Pipeline Manager Chooses a Rate Plan for Subscription Services

Member services use the rate plan associated with the products and discounts owned by the subscription service. If the subscription service and a member service both have products and discounts, the member service's products and discounts are used to rate the member service usage.

For more information, see "How Pipeline Manager Chooses a Rate Plan" in *BRM Configuring Pipeline Rating and Discounting*.

## How Service Status Changes Affect Subscription Services

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 subscription's member services unless the status of a member service was previously changed independently.

- When you inactivate a subscription service, all of its member services are inactivated.
- When you close a subscription service, all of its member services are closed.
- When you reactivate a subscription service that has been inactivated or closed, all of 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 previously closed member service.

Closing a service cancels all products and discounts owned by that service. If you reactivate a service and want to restore canceled products and discounts, you must repurchase those products and discounts for the service.

### About Setting Up Corporate Accounts That Use Subscription Services

When setting up a corporate account that uses subscription services, rating performance is affected if the company has many employees whose services share account balances.

For corporate accounts, it is better to set up an account hierarchy in which the company account is the parent account and the employees own child account that include their own subscription and member services. This way, the charges are rolled up to the corporate account at billing time, so rating performance is less affected.

## About Creating and Managing Subscription Services

A subscription consists of a set of services that you group together by assigning the subscription service and adding any number of member services. Setting up subscription service groups includes these tasks:

- About Setting Up a Subscription Service
- About Configuring a Subscription Service Group
- About Adding Deals to Subscription Services
- About Setting Up Balance Groups for Subscription Services
- About Setting Up Subscription-Level ERAs
- About Modifying Subscription Services
- About Canceling a Subscription Service
- About Transferring a Subscription Service to Another Subscriber

### About Setting Up a Subscription Service

A subscription service is a **/service** storable object in the BRM database. You can create a new 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.

For example, to use subclasses for member services, you might use the **/service/telco/gsm** storable object as the subscription service and **/service/telco/gsm/telephony** and **/service/telco/gsm/sms** objects as member services.

To create custom storable objects that represent subscription services, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

### About Configuring a Subscription Service Group

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 be used again as a member service in the same subscription group.

**Note:** Although you can create a subscription group with optional member services, the optional member services cannot be purchased separately.

You can configure a subscription service group by using only the services included in the plan.

You can create subscription services in two ways:

- When creating a plan in Pricing Center. When customers purchase the plan, the member service objects are created and associated with the subscription service object. Additional member services can be added during customer registration.

  To create subscription services by using Pricing Center, see "Creating a Subscription Service Group".

  Subscription groups are configured in Pricing Center; the deals associated with the subscription and member services can be purchased in Customer Center.

- If you implement your own customer relationship management (CRM) system, customer service representatives (CSRs) can dynamically set up subscription services or add member services to an existing subscription service when registering customers. To create subscription service groups when registering customers, you implement BRM opcodes in your CRM system.

## About Adding Deals to Subscription Services

You can add deals to the subscription service and to member services. A deal owned by the subscription service is a subscription-level deal. That is, the products in the deal are available for rating all services in the subscription group.

You associate deals with subscription services when you add deals to plans in Pricing Center.

A CSR can also add a deal to a subscription service or member service that a customer has purchased in Customer Center.

## About Setting Up Balance Groups for Subscription Services

When using Pricing Center to create subscription services in plans, you can assign all member services to the subscription service's balance group, create a balance group for each individual service in the group, or associate groups of services with different balance groups.

When you create a balance group for each member service in Pricing Center, if you have set up any credit limits, credit floors, and credit thresholds for the subscription's service, those values will be set in the balance group of each member service. You can later change these in the service.

When setting up subscription services during customer registration, you specify which balance group each service in the group should use.

You can change the credit limit on a balance group for a subscription service in Customer Center.

## About Setting Up Subscription-Level ERAs

To set up a subscription-level extended rating attributes (ERAs), you configure a service-level ERA for the subscription service. You set up service-level ERAs for a subscription service the same way that you set up service-level ERAs for an account. See "About Provisioning Tags for Telco Services" in *BRM Telco Integration*.

Any member service that does not have its own 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 usage.

After setting up ERAs, you configure them for a specific customer during customer registration.

## About Modifying Subscription Services

You can perform the following actions to modify an existing subscription's services:

- Add a member service to a subscription group.
- Add deals to a service in a subscription group.
- Change the status of a service in a subscription group.

To modify a subscription's services when setting up services in Pricing Center, see the Pricing Center Help.

To modify the subscription and member services that a customer has purchased in Customer Center, see "Managing Services". Otherwise, to modify a subscription's services after customers have purchased them, you implement BRM opcodes in your CRM system.

## About Canceling a Subscription Service

A subscription service can be canceled at any time during the billing cycle. You can also specify to bill the account for the service immediately upon cancellation.

In BRM, a subscription service is canceled when:

- The account that owns the subscription service is closed.
- The subscription service has expired.
- The subscription service is canceled by a CSR; for example, in Customer Center.

To perform a subscription service cancellation, BRM does the following:

- Closes the subscription service and all of its member services. See "About Service Status When a Subscription Service Is Canceled".
- Cancels the products and discounts associated with the subscription service and its member services. See "Effect of Canceling a Subscription Service on Products and Discounts 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 Resource Sharing Groups Owned by the Service".
- Bills the account, if specified. See "About Billing on Subscription Service Cancellation".

Your CSRs can use Customer Center to cancel subscription services as they cancel other services. To cancel subscription services with your client application, you implement BRM opcodes. For more information, see "Canceling a Subscription Service".

### About Service Status When a Subscription Service Is Canceled

When a subscription service is canceled, all of its member services are canceled as well. The status of the subscription service and its member services are 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 the member services are set to PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE, which specifies that the member service's status was changed due to the status change of the subscription service.

- The purchase, usage, and cycle end dates for the subscription service and all of its member services are set to the cancellation date.

For more information, see "How Service Status Changes Affect Subscription Services".

### Effect of Canceling a Subscription Service on Products and Discounts Owned by the Service

Canceling a subscription service cancels all the product and discount instances owned by the service and its member services.

- The status of the product and discount instances owned by the subscription service and its member services are set to *canceled*.
- The purchase, usage, and cycle end dates for the canceled product and discount instances are set to the subscription service cancellation date.

### Effect of Canceling a Subscription Service on Resource Sharing Groups Owned by the Service

A subscription service and its member services can own a resource sharing group, be members of a resource sharing group, or both.

When a subscription service is canceled, BRM does the following:

- For each sharing group member, it deletes the sharing group's balance group from the group member's ordered balance group list.
- Removes the subscription service and its member services from any sharing groups in which they are members.

For more information on sharing groups, see "About Resource Sharing Groups" in *BRM Managing Accounts Receivable*.

## About Transferring a Subscription Service to Another Subscriber

You can transfer a subscription service from one subscriber to another at any time during the billing cycle.

Transferring a subscription service also transfers all member services to the new subscriber's account. The member service objects are updated to reference the new subscriber's account object.

**Important:** When you transfer a subscription service from one account to another, its balance group is also transferred. If the subscription service (or one of its member services) uses an account-level balance group, however, that balance group cannot be transferred. (Transferring the account-level balance group would leave the original account without a balance group, which is not allowed.) As a result, the subscription service itself cannot be transferred. To avoid this problem, the subscription service and member services should use service-level balance groups. You can define a balance group for the subscription service and associate all the member services with that balance group. See "About Creating and Managing Subscription Services".

The following functions are not supported for subscription service transfer:

- Transferring a subscription service across different brands.

- Backdating of a subscription service transfer.

Each time a subscription service is transferred, BRM stores information about the old account and the date and time when the transfer occurred in a *transfer list*.

The transfer list is stored in the **/service** object. Information stored in the transfer list is used by BRM to rate and record any delayed events and call details records (CDRs) and to bill for usage created prior to the service transfer for the old subscriber's account.

To transfer subscription services, you implement BRM opcodes in your client application.

### Transferring Subscription Services Between Accounts with Different Statuses

You can transfer a subscription service from a closed account to an account that is active, as well as transfer a subscription service from an active account to an account that is closed. When a subscription service is transferred from a closed account to an active account, the subscription service's status remains closed after the transfer. You must manually activate the subscription service. When a subscription service is transferred from an active account to a closed account, the subscription service's status remains active after the transfer. In this case, you must manually activate the closed account.

You can transfer pending scheduled actions associated with an existing subscription when you transfer the subscription to a different account. All the **/schedule** objects corresponding to the member services (including the parent service) are transferred to the target account. Completed actions (that is, **/schedule** objects with **Done** status) are not transferred.

If you want to disallow the transfer of a subscription service to an account that is closed, you can do this by using event notification and writing a custom opcode to generate an error when the subscription service transfer event occurs. For more information on using event notification, see "Using Event Notification" in *BRM Developer's Guide*.

### About Transferring Subscription Services Between Schemas

To transfer a a subscription service between accounts stored in multiple schemas, you can do one of the following:

- Migrate the accounts to the same schema and then transfer the service.

  For example, if you have:

  – Account A stored in Schema 1

  – Account B stored in Schema 2

  To transfer a subscription service from Account B to Account A, you must move Account B to Schema 1, and then transfer the subscription service to Account A.

  See "Transferring Subscription Services Between Accounts in the Same Schema" for information about transferring subscriptions services in the same schema.

  BRM provides the Account Migration Manager (AMM), which can be used to migrate accounts between database schemas.

  **Important:** AMM is an optional feature. Contact your BRM account manager for more information on using AMM.

- If you do not want to migrate the accounts to the same schema, enable the **RecreateDuringSubscriptionTransfer** business parameter. See "Transferring Subscription Services Between Accounts in Multiple Schemas" for more information.

### About Transferring Objects Associated with a Subscription Service

When a subscription service is transferred, objects associated with the service and member services are transferred to the new subscriber's account.

In BRM, a service can be associated with the following objects:

- Service profiles. See "About Transferring Service Profiles Associated with a Subscription Service".
- Devices and associated phone numbers for the service. See "About Transferring Devices Associated with a Subscription Service".
- Products and discounts. See "About Transferring Products and Discounts Associated with a Subscription Service".
- Service resource sub-balances and billing information. See "About Transferring Service-Level Balance Groups during Subscription Service Transfer".
- Service groups. See "How a Subscription Service Transfer Affects Charge and Discount Sharing Groups Owned by the Service".

#### About Transferring Service Profiles Associated with a Subscription Service

In BRM, you can create service profiles to store additional information about the service. A service profile object is associated with a service and the account that the service belongs to. When a subscription service is transferred to a new subscriber's account, the profile objects associated with the subscription service and the member services are updated to reference the new subscriber's account object.

For more information on profiles, see "About Collecting Nonstandard Account Information".

#### About Transferring Devices Associated with a Subscription Service

In BRM, 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 subscription service is transferred to a new subscriber's account, the devices associated with the subscription service and the member services are updated to reference the new subscriber's account object.

For more information on devices, see "Managing Devices with BRM" in *BRM Developer's Guide*.

#### About Transferring Products and Discounts Associated with a Subscription Service

BRM rates events by using the product 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 product or a discount can be purchased and cycle fees and usage charges can be applied.

When a subscription service is transferred, BRM cancels the products and discounts associated with the subscription service and member services for the old subscriber 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 products and discounts for the old subscriber's account.

**Important:** For delayed events to be rated and discounted for the old subscriber's account, the canceled product and discount instances must persist in the account. For more information, see "Rating Delayed Events for Canceled Products" and "Rating Delayed Events for a Canceled Discount".

Another instance of the products and discounts associated with the subscription service and member services is added to the new subscriber's account with the purchase, cycle, and usage start dates set to the transfer date. The end dates are set to the old end dates. Events generated by the subscription service after the transfer date are then rated and discounted for the new subscriber's account.

**Note:**

- To discount any charges associated with the products, such as the prorated cycle fee amount, the discount instances are added to the new subscriber's account before the product instance.
- When a subscription service is transferred, the subscriber accounts are not charged with any purchase or cancellation fees that are generally applied when a product or discount is purchased or canceled.
- When a subscription service is transferred, customized products are transferred along with their base product.

For more information on customer's products and discounts, see "Managing Customers' Services and Products".

### About Transferring Sponsored Products

When a subscriber's account owns sponsored products, the sponsored products are associated with the sponsor group that the subscriber is a member of. The balance impacts for the sponsored products are applied to the sponsor group owner account.

When a subscription service is transferred, one of the following occurs for sponsored products:

- If the new subscriber account is not a member of any sponsor group, reference to the sponsor group is removed from the product instance.
- If the new subscriber account is a member of another sponsor group and the same product is sponsored, reference to the sponsor group is changed to the new sponsor group in the product instance.
- If the product was not sponsored for the old subscriber account but is sponsored for the new subscriber account, a reference to the new sponsor group is added to the product instance.

For more information on sponsorship, see "About Resource Sharing Groups" in *BRM Managing Accounts Receivable*.

### How a Subscription Service Transfer Affects Charge and Discount Sharing Groups Owned by the Service

Charge sharing and discount sharing groups associated with the subscription service or member services are not transferred when the subscription service is transferred to a new subscriber's account.

When a subscription service or member service owns a discount sharing or charge sharing group, the group is deleted when the service is transferred, and the group's balance group is deleted from the ordered balance group list for each member in the group. You need to re-create the groups for the service after the service has been transferred into the new subscriber's account.

When a subscription service or member service is a member of a charge sharing or discount sharing group, the service is deleted from the group when it is transferred, 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 has been transferred.

For more information on deleting a sharing group or deleting a sharing group member, see "About Sponsor Groups" in *BRM Developer's Guide*.

### About Transferring Service-Level Balance Groups during Subscription Service Transfer

A subscription service can have its own service-level balance group. Each member service can also have its own balance group. The balance group is associated with the subscriber account's bill units.

When a subscription service is transferred, the balance groups associated with the subscription service and the member services are transferred to the new subscriber's account. Either the balance groups are added to an existing bill unit in the new subscriber's account or a new bill unit is created for the service only.

**Note:**

- You cannot transfer subscription services that use account-level balance groups. When you transfer a subscription service from one account to another, its balance group is also transferred. If the subscription service (or one of its member services) uses the account-level balance group, however, that balance group cannot be transferred. (Transferring the account-level balance group would leave the original account without a balance group, which is not allowed.) As a result, the subscription service itself cannot be transferred. To avoid this problem, subscription services should use service-level balance groups. See "About Creating and Managing Subscription Services".
- BRM does not allow movement of a balance group from one bill unit to another when the balance group's bill unit has unallocated payments or adjustments, open refunds, or unresolved disputes. However, this restriction does not apply when a balance group is moved to another account as a result of a subscription service transfer.

### How Events Are Assigned to Bill Items after a Subscription Service Transfer

Each billable 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 subscriber's account.

When a subscription service is transferred, BRM creates new bill items for the new subscriber's 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 tied to the bill unit (**/billinfo**) that is referenced by the service's balance group in the new

subscriber's account. As a result, charges for the new events are applied to the new subscriber's account.

Delayed events and CDRs are assigned to the bill items in the old subscriber's account. The bill items are associated with the old subscriber's bill unit; therefore, charges for the bill items are applied to the old subscriber's account.

### About Transferring Non-Currency Resources during Subscription Service Transfer

When a subscription service is transferred from one subscriber's account to another, available non-currency resources that were granted by cycle forward fees are prorated and stored in two sub-balances with different validity periods.

- *Original sub-balance:* By default, resources in this sub-balance are valid from the start of the original account's current billing cycle to the transfer date.
- *New sub-balance:* By default, resources in this sub-balance are valid from the transfer date to the end of the new account's current billing cycle.

Before a subscription service is transferred, a resource is stored in a single sub-balance. After a subscription service is transferred, each resource is prorated and stored in two sub-balances: the original sub-balance and a new sub-balance.

> **Note:** To create separate sub-balances with different validity periods, the balance impact that grants the free resource must have a non-zero relative cycle start date. If the balance impact's relative cycle offset in the cycle forward or cycle arrears rate plan is set to 0, a new sub-balance is not created. Instead, the resources are added to the original sub-balance and are valid forever.

Rating and discounting impact the resources in the original sub-balance for delayed events generated by the service prior to the transfer date. Resources in the new sub-balance are consumed for new events generated by the subscription service after the transfer date.

By default, the original sub-balance always expires on the transfer date. Because events generated by the new account occur after the transfer date, the new account cannot consume free resources from the original sub-balance. You can extend the validity period of the original sub-balance so that its free resources can be consumed by the new account. See "About Extending Sub-Balance Validity When a Subscription Service Is Transferred".

### About Extending Sub-Balance Validity When a Subscription Service Is Transferred

When a subscription service is transferred from one account to another, non-currency resources are prorated and stored in sub-balances with different validity periods. See "About Transferring Non-Currency Resources during Subscription Service Transfer".

To extend the validity period of the original sub-balance, you set the **sub_bal_validity** business configuration parameter to one of the following options:

- **Cut:** This is the default. This option sets the validity period from the start of the original account's current billing cycle to the service transfer date.
- **Maintain:** This option retains the original end date of the bucket. This is true even in the case of a product cancellation or line transfer.
- **Align:** This option sets the validity period from the start of the original account's current billing cycle to the end of the new account's current billing cycle.

**Note:**

- Setting the **sub_bal_validity** business configuration parameter extends the validity period of the original sub-balance only. The resources in the new sub-balance are always valid from the service transfer date to the end of the new account's billing cycle.
- Extending the validity period of the original sub-balance extends only the time the free resources in the bucket are accessible and does not change the resource proration. The free resources are prorated based on the service transfer date. For more information, see "How Cycle Fees Are Prorated When a Subscription Service Is Transferred".
- If you do not configure the **sub_bal_validity** parameter, BRM sets the original sub-balance to expire on the service transfer date, and the new account cannot access this sub-balance.

To configure sub-balance validity, see "Configuring Sub-Balance Validity for Subscription Service Transfer".

### How Non-Currency Resources Are Consumed When a Subscription Service Is Transferred

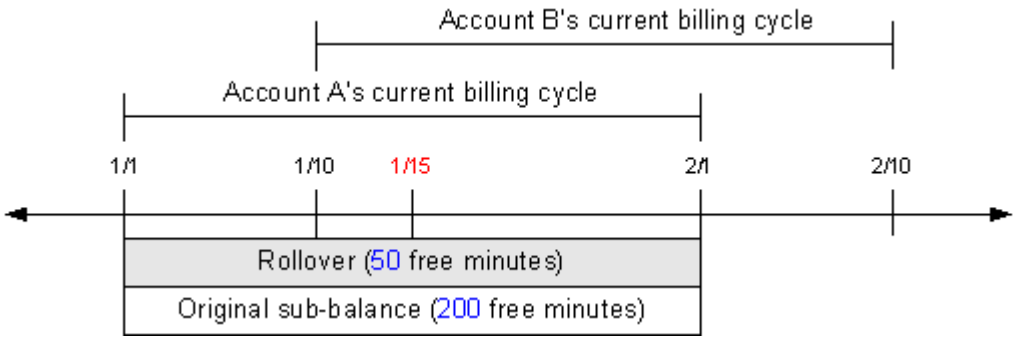
When a subscription service is transferred from one account to another, non-currency resources are prorated and stored in sub-balances with different validity periods. See "About Transferring Non-Currency Resources during Subscription Service Transfer".

The original account always consumes free resources from the original sub-balance because the events for this account occur before the service transfer date. The new account, however, can consume resources 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 "About Extending Sub-Balance Validity When a Subscription Service Is Transferred".

The following examples demonstrate how non-currency resources are consumed after a subscription service is transferred.

On January 1, Account A has a rollover balance of 50 free minutes, rolled over from the previous cycle, and a new grant of 200 free minutes. Subscription service A is later transferred from Account A to Account B on January 15. A timeline for the subscription transfer, including, free minute balances, is shown in Figure 14–10:

***Figure 14–10 Example Timeline***

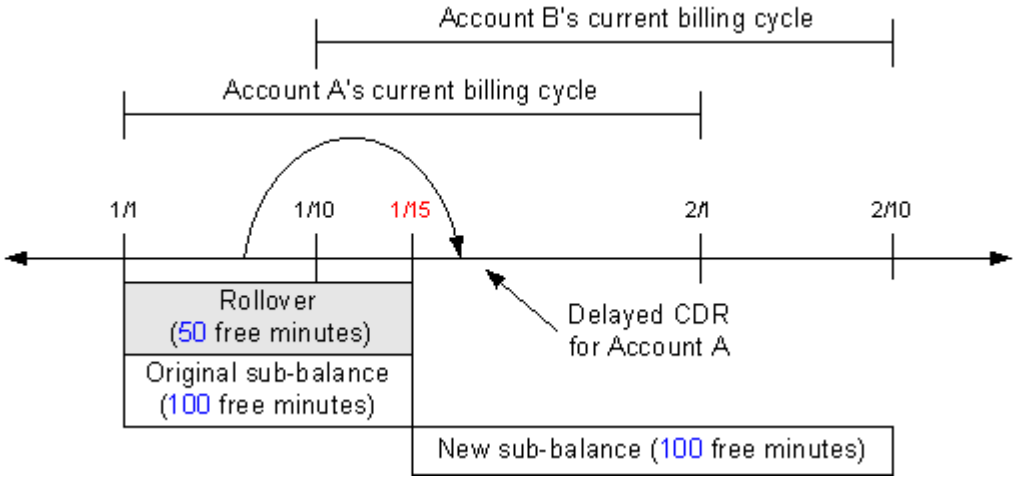


### Example 1

When **sub_bal_validity** is set to **Cut**, the rollover and original sub-balance buckets' **valid_to** dates are set to the transfer date. The resources in the original sub-balance are prorated and a new sub-balance is created with 100 free minutes.

A delayed CDR for Account A is recorded on January 17 for 200 minutes. When the CDR 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 CDR 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 14–11 illustrates this example.

***Figure 14–11 Example 1***

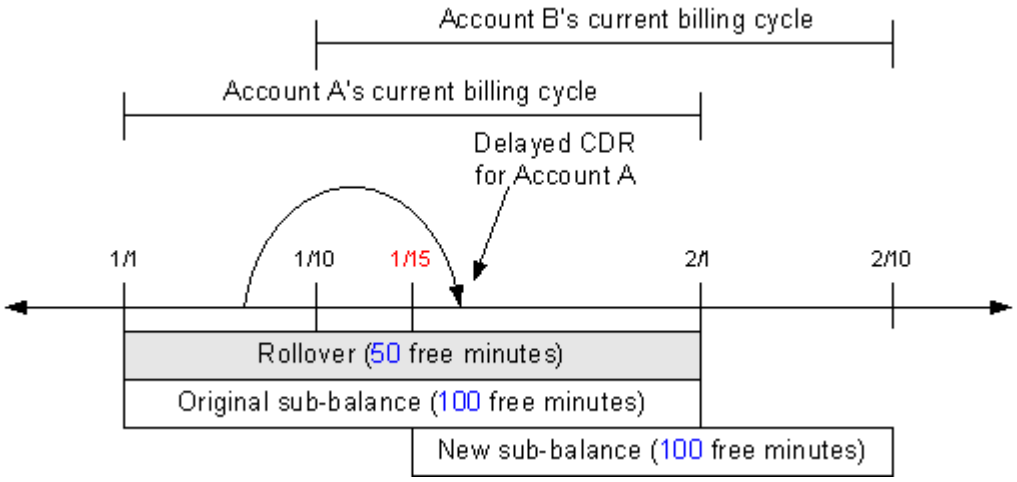



### Example 2

When **sub_bal_validity** 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 CDR for Account A is recorded on January 17 (after the service transfer date) for 200 minutes. When the CDR 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 CDR 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 14–12 illustrates this example.

*Figure 14–12 Example 2*

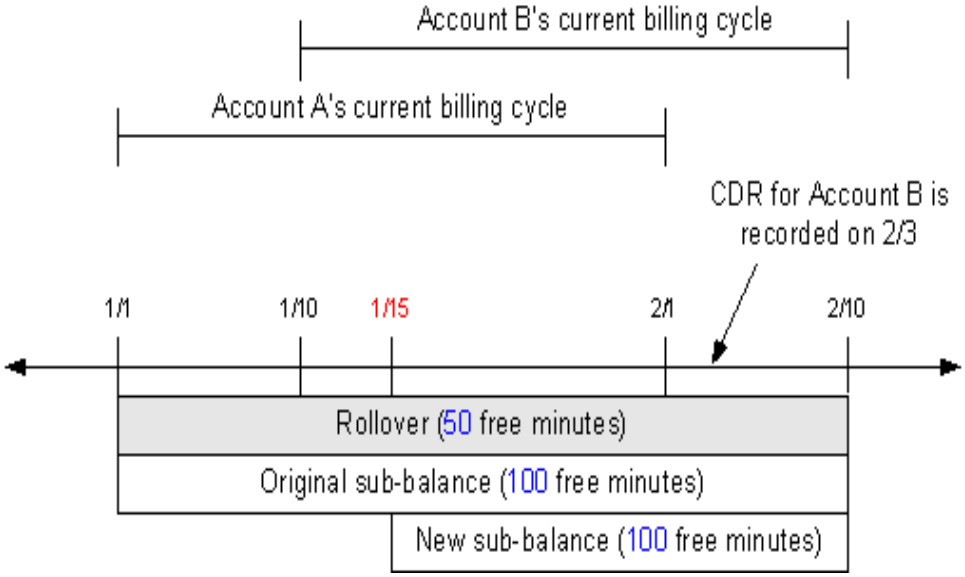



### Example 3

When **sub_bal_validity** 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.

A CDR 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 14–13 illustrates this example.

*Figure 14–13 Example 3*



### How Non-Currency Resources Are Rolled Over When a Subscription Service Is Transferred

You can configure rollover so that non-currency resources associated with cycle forward fees are rolled over for use in future cycles. For more information, see "About Rollovers" in *BRM Setting Up Pricing and Rating*.

When a subscription service is transferred, the free resources in the rollover and original sub-balance buckets are rolled over when **sub_bal_validity** is set to **Maintain** or **Align**.

> **Note:** The rollover and original sub-balance buckets are not rolled over when **sub_bal_validity** is set to **Cut**.

The following example demonstrates how unused resources are rolled over when a subscription service is transferred to another account. In this example:

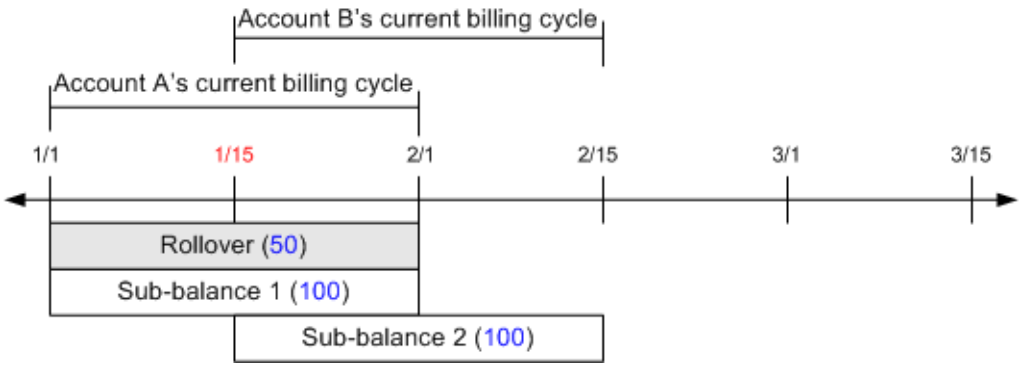
- The subscription service grants 200 free minutes valid from January 1.
- The subscription service is transferred to the new subscriber's account on January 15.
- The billing cycle for the original account starts on the first of each month.
- The billing cycle for the new account starts on the 15th of each month.
- The **sub_bal_validity** business configuration parameter is set to **Maintain**.
- Rollover minutes from the previous cycle are 50.

The rollover rules are:

- A maximum of 50 free minutes can be rolled over to the next cycle.
- The maximum number of rollover cycles is 1.
- The maximum number of free minutes that can be rolled over from previous months is 100.

On January 1, the old subscriber's account is granted 200 free minutes for the month of January. On January 15, the subscription service is transferred to the new subscriber's 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 14–14 illustrates this example.

***Figure 14–14 Example 4a***

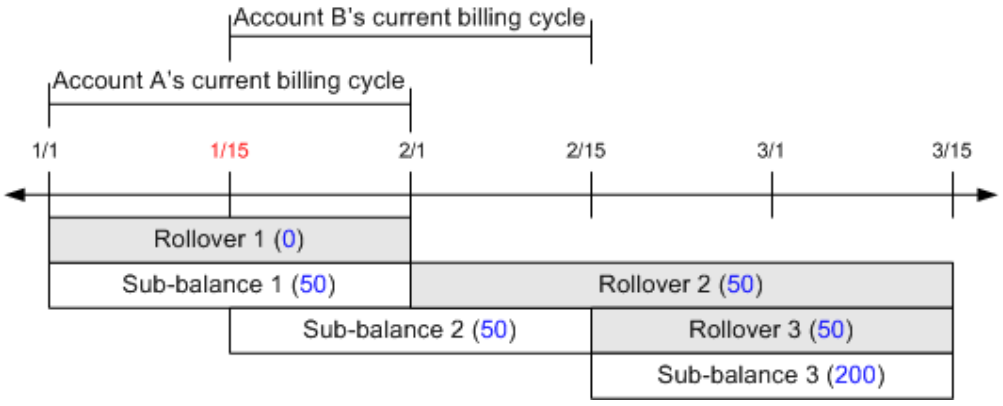


When **pin_bill_day** is run on February 1, 50 minutes from Sub-balance 1 are rolled over. Because Sub-balance 1 is valid for a partial cycle (from January 15 to February 1 instead of January 15 to February 15), the resources are rolled over for one entire cycle. As a result, the Rollover 2 bucket is 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 new account is granted an additional 200 minutes for the new cycle (Sub-balance 3).

Figure 14–15 illustrates the billing timeline associated with this example.

***Figure 14–15 Example 4b***




Because the Rollover 2 and Rollover 3 buckets are valid after the transfer date, only the new account can access the resources in these buckets. The old account cannot consume the free minutes from these buckets.

### How Cycle Fees Are Prorated When a Subscription Service 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 subscription service transfer is not the same as canceling the product in the old account and purchasing it for the new account. Instead, a service transfer moves the service from one account to another. When a subscription service is transferred, the subscriber accounts are not charged with the product purchase or cancellation fees and therefore the purchase and cancellation proration are also not applied that are generally applied when a product or discount is purchased or canceled.

Consequently, when a subscription service is transferred during a billing cycle, cycle fee proration is enforced based on the transfer date to charge the old account for the fees before the transfer and the new 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 old subscriber's account, and the prorated cycle fees from the transfer date to the end of the billing cycle are applied to the new subscriber's account.

For more information on how cycle fees are prorated, see "About Proration" in *BRM Configuring and Running Billing*.

### How Billing Time Discounts and Folds Are Applied When a Subscription Service Is Transferred

Billing time discounts and folds are applied at the end of the accounting cycle and are based on resources used during the cycle. In BRM, resource balances are stored and tracked in account-level or service-level balance groups.

When a subscription service is moved to a new subscriber's account, the service-level balance group is transferred to the new subscriber's account. As a result, billing-time folds and discounts are applied to the new subscriber's account. See "About Transferring Service-Level Balance Groups during Subscription Service Transfer".

For more information on billing time discounts, see "About Implementing Discounts" in *BRM Configuring Pipeline Rating and Discounting*.

### About Rating Delayed Events When a Subscription Service Is Transferred

Delayed events generated by a subscription service that occur prior to the subscription service transfer are billed to the old subscriber's account. See "How Events Are Assigned to Bill Items after a Subscription Service Transfer".

BRM rates events by using the product and discount information associated with the service for which the events are generated. When a subscription service is transferred, BRM stores the original account's products and discounts to rate and discount delayed events generated by that account. See "About Transferring Products and Discounts Associated with a Subscription Service".

When rating delayed CDRs for the original account, the original account can be charged for usage when all of the free resources have been consumed by the new account. This can occur when the validity period of the original sub-balance is extended beyond the subscription service transfer date. See "How Non-Currency Resources Are Consumed When a Subscription Service Is Transferred".

> **Important:** To achieve correct rating results when rating delayed events, you must rerate the events for both the original and new account in the correct sequence. For more information on rerating, see "About Rerating Events" in *BRM Setting Up Pricing and Rating*.

### How a Subscription Service Transfer Affects Account Migration

When an account selected for rerating is associated with a subscription service transfer, rerating requires both the original account and the account to which the service was transferred to reside in the same database schema. For this reason, the Account Migration Manager (AMM) does not allow the original account or the account to which a service was transferred 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 subscription service from Account A to Account B, both accounts must reside in the same schema. You must move Account B to Schema A or Account A to Schema B, and then transfer the service to Account B.

If, after the service transfer, you want to migrate Account A or Account B to Schema C, AMM does not allow the migration because rerating requires the original account and the account to which the service was transferred to reside in the same schema. If you

want to transfer the service again to Account C, you must move Account C to Schema A, and then transfer the service.

### Configuring Sub-Balance Validity for Subscription Service Transfer

To extend the validity period of the original sub-balance when a subscription service is transferred, you change the **sub_bal_validity** business configuration parameter. This parameter takes one of the following values:

- **Cut**
- **Maintain**
- **Align**

**sub_bal_validity** is a system-wide parameter. By default, this parameter is set to **Cut**. Changing the value of this parameter affects the sub-balance validity period for any subscription service transfer that occurs after the change. For more information, see "About Extending Sub-Balance Validity When a Subscription Service Is Transferred".

**Caution:** If you change the **sub_bal_validity** parameter (for example, from **Align** to **Maintain**), 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.

**Note:** If you do not configure the **sub_bal_validity** parameter, BRM follows the default implementation by setting the original sub-balance to expire on the service transfer date.

To extend the validity period of the original sub-balance, you modify a field in the **/config/business_params** object created during BRM installation. You modify the **/config/business_params** object by using the **pin_bus_params** utility. For more information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

1. Go to the *BRM_Home***/sys/data/config** directory, where *BRM_Home* is the directory in which you installed BRM.
2. Run the following command, which creates an editable XML file from the **billing** instance of the **/config/business_params** object:

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

   This command creates the XML file named **bus_params_billing.xml.out** in your working directory. To place this file in a different directory, specify the path as part of the file name.
3. Open the **bus_params_billing.xml.out** file.
4. Search the XML file for the following line:

   ```
   <SubBalValidity>Cut</SubBalValidity>
   ```

5. Change **Cut** to one of the following:
   - **Maintain** if you want to extend the validity of the original sub-balance to the end of the original account's current billing cycle.
   - **Align** if you want to extend the validity of the original sub-balance to the end of the new account's current billing cycle.

6. Save the file as **bus_params_billing.xml**.
7. Run the following command, which loads this change into the **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_billing.xml
   ```

   where *PathToWorkingDirectory* is the directory in which the **bus_params_AR.xml** file resides.

   **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 billing configuration.

   **Note:** To run this command from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

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

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using the **testnap** utility. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

9. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
10. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

## Mapping Subscription Services in the Pipeline Manager Database

To rate subscription services in a pipeline, the data that defines the subscription service object must be mapped in the Pipeline Manager database. Sample mapping is provided by BRM for some service objects. However, if you set up a service object for the subscription service that is not already configured in the Pipeline Manager database, you must map that service to the data that defines it.

Use Pricing Center to map the subscription service in the following Pipeline Manager database tables:

- IFW_SERVICE specifies the pipeline rating service code to real-time rating service code mapping; for example, map **GSM** to **/service/telco/gsm**.
- IFW_SERVICE_MAP specifies the external data used to determine the internal service code. You define the mapping when you set up service code mapping.
- IFW_REF_MAP specifies the service-type-to-event-type mapping so that the services in your price list are associated with usage events. This database table also specifies the prefix for the data that identifies an account; for example, MSISDN or IMSI.

**Important:** Be sure to map a corresponding delayed event type to your subscription service object. For example, if you map to the usage event type **/event/session/telco/gsm**, also map to **/event/delayed/session/telco/gsm**. The delayed event type is used by Rated Event (RE) Loader when loading the rated events into the BRM database. For more information, see "Understanding Rated Event Loader" in *BRM Configuring Pipeline Rating and Discounting*.

If the delayed event type does not exist in the BRM database, you must create one. See "Adding New Event Types for RE Loader to Load" in *BRM Configuring Pipeline Rating and Discounting*.

- IFW_ALIAS_MAP specifies which data to use for looking up an account in order to get the parameters for rating usage. You define the alias mapping when you set up EDR container fields. For more information, see "Modifying and Loading the EDR Container Description" in *BRM Configuring Pipeline Rating and Discounting*.

  To define the alias mapping by using Pricing Center:

  a. From the **Pipeline Setup Toolbox**, select **EDR - EDR Container Description**.
  b. In the EDR Container Description dialog box, select the **ALL_RATE** sample container description and click **Edit**.
  c. Click the **Alias Map** tab and enter the alias data information for the subscription service.

For more information on mapping pipeline data, see "Setting Up Pipeline Price List Data" in *BRM Configuring Pipeline Rating and Discounting*.

# About Billing for Subscription Service Usage

To bill for a subscription, all balance groups in the subscription group must be associated with a *bill unit* (**/billinfo** object). A bill is produced for each bill unit.

**Important:** 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 subscription or use the account's bill unit (the AR bill unit). If you use the account's bill unit, charges for any service owned by the account that are not part of the subscription are also included in the bill.

You use 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 with Your Custom Application" in *BRM Configuring and Running Billing*.

## About Creating Multiple Bills for a Subscription

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.

When member services have their own balance groups, you can create separate bills for those services by associating them with a new bill unit. Charges for any member service that does not have its own balance group are rolled up to the subscription service's bill unit and are included in the subscription service's bill.

## About Billing for Multiple Subscriptions

When a customer owns multiple subscriptions, you can bill for each subscription separately or include charges for all subscriptions in the same bill. To create one bill for all subscriptions, associate the balance groups in all the subscription groups with the same bill unit.

## About Billing on Subscription Service Cancellation

Normally, you bill a subscriber's 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 subscriber's account immediately, you specify the Bill Now option when cancelling the subscription. See "About Bill Now" in *BRM Configuring and Running Billing*.

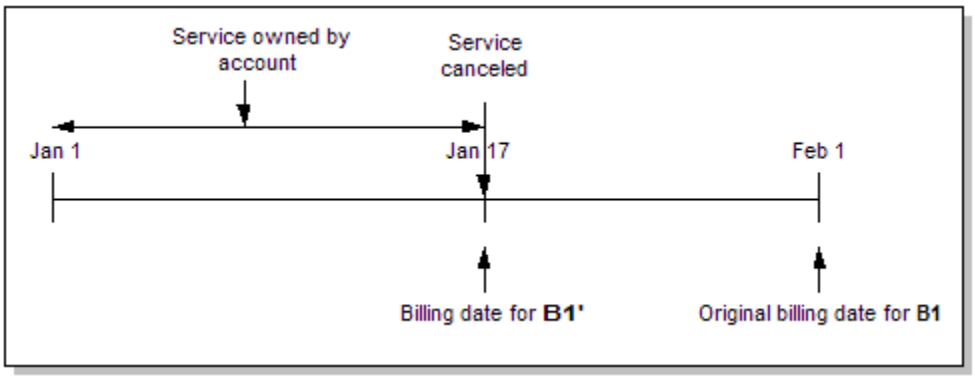
**Note:** If the Bill Now option is not specified, the service is billed during the regularly scheduled billing time.

When the Bill Now option is specified, the subscriber's 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.

In Figure 14–16, the subscription service is canceled on January 17 with the Bill Now option specified. Bill **B1'** includes all current charges between January 1 and January 17. Bill **B1** includes all delayed charges for the service recorded after the cancellation date.

***Figure 14–16 Example 5***




When the Bill Now option is specified, BRM bills the bill units (**/billinfo** objects) 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 are also billed.

If you do not want other services in the bill unit to be billed immediately, do one of the following:

- Wait for the regular billing day, and bill the canceled subscription service along with the other services in the bill unit.
- Customize the PCM_OP_BILL_POL_GET_PENDING_ITEMS policy opcode to include only the pending bill items for the subscription service that was canceled.

You can extend your customer management application to create a Bill Now for a specific service. For more information, see descriptions of the following opcodes:

- PCM_OP_BILL_CREATE_SPONSORED_ITEMS
- PCM_OP_BILL_MAKE_BILL_NOW
- PCM_OP_BILL_POL_GET_PENDING_ITEMS

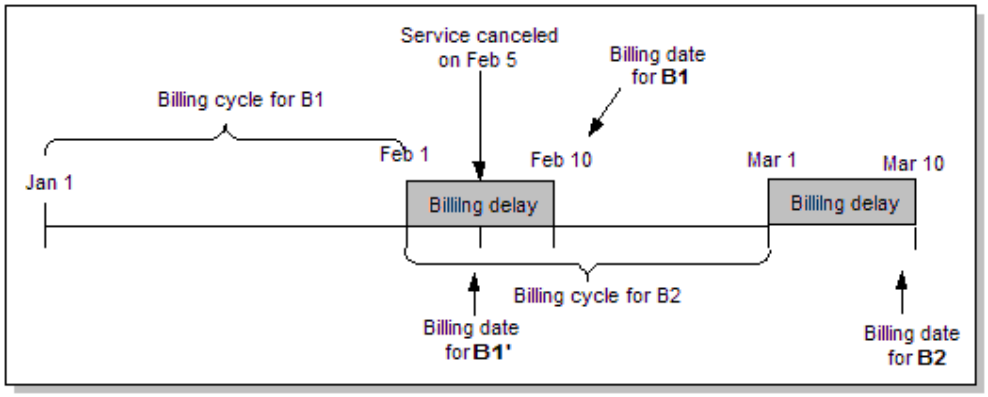
### How an Account Is Billed When a Subscription Service Is Canceled during the Billing Delay Period

When a subscription service is canceled during the billing delay period with the Bill Now option specified, the subscriber's account is billed immediately for all current charges that includes charges from the current billing cycle and the previous billing cycle.

Delayed events recorded after the cancellation date are billed on the regularly scheduled billing date.

In Figure 14–17, 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 14–17 Example 6***




## About Billing a Subscription Service after It Is Transferred to Another Subscriber

A subscription service can be transferred at any time during the billing cycle.

At the time of billing, any cycle fees associated with the service are prorated. The old subscriber's account is billed for the prorated amount until the transfer date. The new subscriber's account is billed for the remainder of the cycle. See "How Cycle Fees Are Prorated When a Subscription Service Is Transferred".

The old subscriber's account is billed for usage charges associated with the service that was incurred during the time the service was owned by the old subscriber's account, including delayed events that occurred prior to the transfer of the service. Events generated after the transfer are billed to the new subscriber's account.

Any remaining non-currency grants, such as free minutes, are transferred to the new subscriber's account. See "About Transferring Non-Currency Resources during Subscription Service Transfer".

# Managing Subscription Services in Your Custom Application

For more information on subscription services, see "About Subscription Services".

## Creating a Subscription Service Group

You can create subscription service groups when creating price plans or when registering customers.

### Creating Subscription Services in a Price Plan

To create a subscription service in a price plan, use the PCM_OP_PRICE_SET_PRICE_LIST opcode and set the following subscription group fields in the input flist:

- Add a PIN_FLD_SERVICES array for each service in the subscription.

  **Important:** The service arrays must start with array element **1**.

- In each service array, set the PIN_FLD_SUBSCRIPTION_INDEX field to specify the index of the array that contains the subscription service.

  **Note:** When you add standalone services to a price plan, you set the PIN_FLD_SUBSCRIPTION_INDEX field to **0**. This value indicates that the service is not a member of a subscription group.

- Add a PIN_FLD_BAL_INFO array to create a balance group for the subscription service. Optionally, specify any subscription-level credit limits or thresholds.
- Add an additional PIN_FLD_BAL_INFO array for each member service that should have its own balance group, and set credit limits and thresholds, if any. A member service that does not have a balance group uses the subscription service's balance group.
- In each PIN_FLD_SERVICES array, set the PIN_FLD_BAL_INFO_INDEX field to specify the index of the array that contains the balance group for that service.

PCM_OP_PRICE_SET_PRICE_LIST validates that:

- The PIN_FLD_SUBSCRIPTION_INDEX field specifies an existing element ID of the PIN_FLD_SERVICES array. If the element ID does not exist, the PIN_PRICE_ERR_INVALID_SUBSCRIPTION_INDEX error is returned.
- Member services from different subscription groups do not share the same balance group. If the PIN_FLD_BAL_INFO_INDEX field for two services that belong to different subscription groups specify the same balance group index array element, the PIN_PRICE_ERR_INVALID_BAL_INFO_INDEX error is returned.

The following example shows an flist with the fields that pertain to subscription service groups. The plan includes three services in a subscription group, one subscription service, and two member services. All services use the subscription service's balance group:

. . .

```
0 PIN_FLD_BAL_INFO     ARRAY [1] allocated 2, used 2 //the subscription service's balance group
     1 PIN_FLD_NAME                  STRING [0] subscription
     1 PIN_FLD_LIMIT                 ARRAY [840] allocated 3, used 3
          2 PIN_FLD_CREDIT_FLOOR           DECIMAL [0] 0.0
          2 PIN_FLD_CREDIT_LIMIT           DECIMAL [0] 500.0
          2 PIN_FLD_CREDIT_THRESHOLDS      INT [0] 0

0 PIN_FLD_SERVICES     ARRAY [1] allocated 3, used 3 // Array of the subscription service
     1 PIN_FLD_SERVICE_OBJ        POID [0] 0.0.0.1 /service/telco/gsm -1 0 //The subscription
service object.
     1 PIN_FLD_SUBSCRIPTION_INDEX     INT [0] 1 //The index of the subscription service. Note that
it references itself.
     1 PIN_FLD_BAL_INFO_INDEX       INT [0] 1 //The index of the subscription service's balance
group array.
      . . .
0 PIN_FLD_SERVICES     ARRAY [2] allocated 3, used 3
     1 PIN_FLD_SERVICE_OBJ          POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0 // Member
service.
     1 PIN_FLD_SUBSCRIPTION_INDEX     INT [0] 1
     1 PIN_FLD_BAL_INFO_INDEX      INT [0] 1
      . . .
0 PIN_FLD_SERVICES     ARRAY [3] allocated 3, used 3
     1 PIN_FLD_SERVICE_OBJ         POID [0] 0.0.0.1 /service/telco/gsm/sms -1 0 // Member service.
     2 PIN_FLD_SUBSCRIPTION_INDEX    INT [0] 1
     3 PIN_FLD_BAL_INFO_INDEX      INT [0] 1
      . . .
```

### Creating Subscription Services When Registering Customers

The **/service** objects for subscription services are created when plans are purchased. This can occur at two times:

- When registering customers, use the PCM_OP_CUST_COMMIT_CUSTOMER opcode.
- When customers purchase a plan for an existing account, use the PCM_OP_CUST_MODIFY_CUSTOMER opcode.

These opcodes call other opcodes to set up a customer's services and create the **/service** objects.

These opcodes set the subscription group relationship fields in the **/service** objects that they create. Services are created in the following order: account-level services, the subscription service, and member services in the subscription group.

To create a subscription service group, set the following fields in the input flist of PCM_OP_CUST_CREATE_CUSTOMER and PCM_OP_CUST_MODIFY_CUSTOMER:

- Add a PIN_FLD_SERVICES array for each service in the subscription group.

  **Important:** The service arrays must start with array element **1**.

- In each service array, set the PIN_FLD_SUBSCRIPTION_OBJ field to specify the Portal object ID (POID) of the subscription service. If the service being created *is* the subscription service, this field and the PIN_FLD_POID field specify the same value.
- Add a PIN_FLD_BAL_INFO array to create a balance group for the subscription service. Optionally, specify any subscription-level credit limit or threshold.

- Add an additional PIN_FLD_BAL_INFO array for each member service that should have its own balance group, and optionally set the credit limit and threshold. A member service that does not have a balance group uses the subscription service's balance group.
- In each PIN_FLD_SERVICES array, set the PIN_FLD_BAL_INFO_INDEX field to specify the index of the array that contains the balance group for that service.

## Configuring ERAs for a Subscription Service

After setting up system-wide ERAs, you assign an ERA to a customer's service and specify customer-specific information; for example, the customer's birthday for a birthday discount ERA.

Service-level ERAs are stored in **/profile/serv_extrating** objects. These objects are typically associated with customer accounts at two times:

- When registering customers, use PCM_OP_CUST_COMMIT_CUSTOMER.
- When customers purchase a plan for an existing account, use PCM_OP_CUST_MODIFY_CUSTOMER.

These opcodes call other opcodes to create or modify the **/profile** objects.

To configure a subscription-level ERA for a specific customer, add a PIN_FLD_PROFILES array to the subscription service's PIN_FLD_SERVICES array on the input flist of the *_CUSTOMER opcodes. In the profiles array, set the POID of the **/profile/serv_extrating** object owned by the service.

To configure the ERA with customer-specific data, write custom code that sets the customer's information in the PIN_FLD_DATA array of the **/profile/serv_extrating** object.

To validate the customer's profile information, you must modify the PCM_OP_CUST_POL_VALID_PROFILE policy opcode. See the opcode documentation for more information.

You can also use the Customer profile opcodes to create and modify **/profile** objects directly. See "Managing and Customizing Profiles".

## Associating a Device with a Subscription Service

Customers use devices, such as a wireless handset, to access their subscription services. The device information, such as device ID, manufacturer, and associated account and service, are stored in a **/device** object. You associate **/device** objects with subscription **/service** objects when you register customers or add plans to customer accounts:

- When registering customers, use PCM_OP_CUST_COMMIT_CUSTOMER.
- When customers purchase a plan for an existing account, use PCM_OP_CUST_MODIFY_CUSTOMER.

These opcodes call other opcodes to create or modify the **/device** objects.

To create a **/device** object and associate it with a subscription service, set the following fields in the subscription service's PIN_FLD_SERVICES array on the input flist of the customer opcodes:

- Specify the POID of the **/device** object in the PIN_FLD_DEVICE_OBJ field of the PIN_FLD_DEVICES array.

- Specify the number associated with the device (such as the IMSI or MSISDN) in the PIN_FLD_ALIAS_LIST array.

You can also use the PCM_OP_CUST_UPDATE_SERVICES opcode to modify **/device** objects. This opcode calls the PCM_OP_DEVICE_ASSOCIATE opcode to associate or disassociate a device with a service.

To directly create or modify **/device** objects, use the PCM_OP_DEVICE* opcodes.

## Adding a Service to an Existing Subscription Service Group

You add a service to a subscription group in the same way that you create a service in a subscription group. Use PCM_OP_CUST_MODIFY_CUSTOMER and set the service information on the input flist. See "Creating a Subscription Service Group".

## Canceling a Subscription Service

For general information about subscription service cancellation, see "About Canceling a Subscription Service".

You can use Customer Center to cancel a subscription service.

For customizations, you can use the PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION opcode directly. This opcode is used to cancel a subscription service and all the member services and to bill the service upon cancellation.

Specify the following information in the PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION input flist:

- Set PIN_FLD_POID to the POID of the account object that owns the subscription service.
- Set PIN_FLD_SUBSCRIPTION_OBJ to the POID of the subscription service to be canceled.
- Set the input flag PIN_FLD_FLAGS to PIN_BILL_FLG_BILL_NOW, to bill the account immediately upon service cancellation.

> **Note:** If the input flag is not set, the account is billed for the service on the regularly scheduled billing date.

PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION performs the following tasks:

1. If the PIN_BILL_FLG_BILL_NOW flag is passed in the input flist, calls the PCM_OP_BILL_MAKE_BILL_NOW opcode for each bill unit (**/billinfo** object) associated with the subscription service.

   > **Note:** Typically, all the member services are associated with the subscription service's bill unit. However, it is possible for member services to be associated with different bill units.

2. Calls the PCM_OP_CUST_UPDATE_SERVICES opcode to update the status of the subscription service and member services. PCM_OP_CUST_UPDATE_SERVICES does the following:
   a. Updates the status of the subscription service and all of its member services to *closed*.

b. Updates the status of all discounts and products associated with the subscription service and member services to *canceled*.

c. Deletes any resource sharing groups owned by the subscription service and the member services, and updates the ordered balance group (**/ordered_balgrp** object) of the group members.

d. Removes the subscription service and member services from any resource sharing group of which they are members.

3. Generates the **/event/audit/subscription/cancel** audit event to record the service cancellation.

If the PIN_BILL_FLG_BILL_NOW flag is specified in the input flist, the service is billed upon cancellation. Otherwise, the service is billed during the regularly scheduled billing time.

If successful, PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION returns the following:

- The POID of the subscription service that was canceled.
- The POID of the **/event/audit/subscription/cancel** audit event that was created to record the cancellation.

## Transferring Subscription Services Between Accounts in the Same Schema

For general information about subscription service transfer, see "About Transferring a Subscription Service to Another Subscriber".

To transfer a subscription service to another subscriber's account, use PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION.

Specify the following information in the PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION opcode input flist:

- The POID of the account object from which the service is transferred.
- The POID of the account object to which the service is transferred.
- The POID of the subscription service to transfer.
- The POID of a bill unit (**/billinfo** object) and the POID of a **/payinfo** object (**/payinfo**):
  - If the service is to be associated with an existing bill unit in the new subscriber's account, specify the POIDs of any existing bill unit and **/payinfo** objects.
  - If the service is to be associated with a new bill unit created for the service only, specify the POIDs of the new bill unit and payinfo objects.
- The POID of an existing payment object (**/payinfo**) for the new subscriber or the POID of a new payment object created for the service only.

To transfer the services, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following for each member service:

1. Sets the account object POID to the new subscriber's account object POID.
2. Creates a transfer list array element and adds it to the service objects.
3. Creates the audit event **/event/audit/subscription/transfer** to record the service transfer.

To transfer the devices, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following for each member service:

1. Finds all the devices owned by the service and sets each device's account object POID to the new subscriber's account object POID.
2. Creates the **/event/audit/subscription/transfer** audit event to record the device transfer.

To transfer the products and discounts, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION cancels the products and discounts owned by the old subscriber's account and creates copies for the new subscriber's account. To do this, it does the following for each member service:

1. Finds all products and discounts associated with the service owned by the existing subscriber's account.
2. For each product instance:
   a. Calls the PCM_OP_SUBSCRIPTION_SET_PRODINFO opcode to set the purchase, cycle, and usage end dates to the transfer date for the old subscriber's account and to apply cycle forward fees.
   b. Creates another instance of the product and assigns it to the new subscriber's account with purchase, cycle, and usage start dates set to the transfer date.

      **Note:** For customized products, the opcode does not separately set end dates in the old account and start dates in the new account. The start and end dates of customized products are adjusted automatically when the dates for their base products are set.

   c. Creates the **/event/billing/product/action/purchase** event to record the product instance transfer.

      **Note:** If the product is sponsored, and the new subscriber's account is not a member of the same sponsor group as the old subscriber's account, the POID of the sponsor group account object is deleted from the product instance.

3. For each discount instance:
   a. Calls the PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO opcode to set the purchase, cycle, and usage end dates to the transfer date for the existing subscriber's account and to apply cycle forward fees.
   b. Creates another instance of the discount and assigns it to the new subscriber's account with purchase, cycle, and usage start dates set to the transfer date.
   c. Creates the **/event/billing/discount/action/purchase** event to record the discount instance transfer.

To transfer the subscription service billing information, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION associates the service balance groups with an existing bill unit in the new subscriber's account or with a new bill unit created for the service. To do this, it does the following:

1. If a **/billinfo** object is specified in the input flist, it sets the PIN_FLD_BILLINFO_OBJ field in the service's **/balance_group** object to reference the **/billinfo** object.

2. If a **/payinfo** object is specified in the input flist, it sets the PIN_FLD_PAYINFO_OBJ field in the service's **/balance_group** object to reference the **/payinfo** object.
3. If a **/billinfo** object is not specified, it calls the PCM_OP_CUST_SET_BILLINFO opcode to create a new **/billinfo** object and sets the PIN_FLD_BILLINFO_OBJ field in the service's **/balance_group** object to the newly created **/billinfo** object.
4. If a **/payinfo** object is not specified, it calls the PCM_OP_CUST_SET_PAYINFO opcode to create a new **/payinfo** object and sets the PIN_FLD_PAYINFO_OBJ field in the service's **/balance_group** object to the newly created **/payinfo** object.

To transfer the subscription service's resource sub-balances, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following:

1. Modifies the service's **/balance_group** object to reference the **/billinfo** object in the subscriber's account.
2. Sets the sub-balance validity based on the **sub_bal_validity** parameter in **/config/business_params**.
3. Creates the **/event/audit/subscription/transfer** audit event to record the balance group transfer.

To transfer pending scheduled actions for a specific service associated with an existing subscription when transferring the subscription to a different account, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION calls the PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION policy opcode.

The **TransferScheduledActions** business parameter must be enabled in order for PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION to call the PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION policy opcode. See "Enabling Transfer of Pending Scheduled Actions during Subscription Transfers".

When a subscription service is transferred to a new subscriber's account, it is deleted from any group of which it is a member. PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following for each member service:

1. Finds the charge sharing or discount sharing groups that the service is a member of and deletes the service from the group.
2. Deletes any ordered balance group associated with the service.

When a subscription service is transferred to a new subscriber's account, any group that it owns is deleted. PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION finds the charge sharing group or discount sharing group that it owns and deletes the group for each member service.

### Enabling Transfer of Pending Scheduled Actions during Subscription Transfers

To enable the transfer of pending scheduled actions during subscription transfers, enable the **TransferScheduledActions** business parameter by doing the following:

1. Go to the *BRM_Home*/**sys/data/config** directory.
2. Run the following command, which creates 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. To place this file in a different directory, specify the path as part of the file name.
3. Open the **bus_params_subscription.xml.out** file.

4. Search for the following line:

   ```
   <TransferScheduledActions>disabled</TransferScheduledActions>
   ```

5. Change **disabled** to **enabled**.
6. Save the file as **bus_params_subscription.xml**.
7. Go to the *BRM_Home***/sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_subscription.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_subscription.xml** resides.

   **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 this file, your changes affect the associated aspects of the BRM subscription configuration.

   **Note:** To run this command from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.

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

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using the **testnap** utility. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.

10. Stop and restart the CM. For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

## Transferring Subscription Services Between Accounts in Multiple Schemas

To transfer a subscription service between accounts stored in multiple schemas, you enable the **RecreateDuringSubscriptionTransfer** business parameter.

**Note:** When the **RecreateDuringSubscriptionTransfer** business parameter is enabled, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION recreates all the objects in the schema to which the subscription services are transferred.

### Enabling Transfer of Subscription Services Between Accounts in Multiple Schemas

To enable transfer of subscription services of an account in a schema to another account in a different schema, enable the **RecreateDuringSubscriptionTransfer** business parameter in the following way:

1. Go to the *BRM_Home*/**sys/data/config** directory.
2. Run the following command, which creates 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. To place this file in a different directory, specify the path as part of the file name.
3. Open the **bus_params_subscription.xml.out** file.
4. Search for the following line:

   ```
   <RecreateDuringSubscriptionTransfer>disabled</RecreateDuringSubscriptionTransfe
   r>
   ```

5. Change **disabled** to **enabled**.
6. Save the file as **bus_params_subscription.xml**.
7. Go to the *BRM_Home*/**sys/data/config** directory, which includes support files used by the **pin_bus_params** utility.
8. Run the following command, which loads this change into the **/config/business_params** object:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_subscription.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_subscription.xml** resides.

   **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 this file, your changes affect the associated aspects of the BRM subscription configuration.

   **Note:** To run this command from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
9. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct.

   See "Using testnap" in *BRM Developer's Guide* for general instructions on using the **testnap** utility. See "Reading Objects by Using Object Browser" in *BRM Developer's Guide* for information on how to use Object Browser.
10. Stop and restart the CM. For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
11. (Multischema systems only) Run the **pin_multidb** script with the **-R CONFIG** parameter. For more information, see "pin_multidb" in *BRM System Administrator's Guide*.

To transfer a subscription service, PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION does the following:

1. Finds all the related services. If the service to be transferred is a subscription service, this opcode finds all the member services and groups.

2. Updates the status of the services to CANCELLED and recalculates charges for all service-level purchased offerings that use PCM_OP_SUBSCRIPTION_SET_PRODINFO or PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO.
3. Creates a flist of all the services to be transferred and saves login information, profiles, and devices.
4. Adds login information to the source services by pre pending the login details with the service POID and generates the **/event/customer/login** event.
5. Calls PCM_OP_CUST_MODIFY_CUSTOMER to create all new service instances in the account in the target schema.
6. Creates the **/billinfo, /payinfo**, and **/balance_group** objects if the source objects do not already have these.

   **Note:** The **/billinfo** object passed in the input must belong to the account in the target schema.

7. Fetches the encrypted password from the service in the source schema and sets it in the service in the target schema and generates the **/event/customer/status** and **/event/customer/login** events.
8. If called from PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER, saves the **/service**, **/balance_group**, and **/billinfo** objects in the NAMED_TRANS_FLIST array.
9. Updates SUBSCRIPTION_OBJ of the newly created instance of the member services in the target schema with the POID of the parent service created in the target schema.
10. Updates the **/device** object with the service and the account in the target schema.

    **Note:** Devices associated with the services to be transferred must not be associated with any other services that are not transferred.

11. Recreates the **/profile** objects in the target schema using CREATE_OBJ that has **effective_t** set to the transfer time.

    **Note:** To cancel the source service instances, the opcode deletes the **/ordered_balgrp** and **/group/sharing/*** relationships and saves the audit copies on the source service instances for processing late CDRs. Sharing group relationships are terminated on the transfer date and the new service instances will not participate in any sharing relationships.

12. Calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE to delete all sharing groups in which the service is a parent or a member.
13. Creates the **/purchased_discount** and **/purchased_product** objects and CYCLE_FORWARD events. For cycle arrears events, calls PRO_FORCED in CYCLE_FEE_FLAGS.
14. Regenerates a package ID for the offerings created in the target schema.

15. Calls PCM_OP_CUST_UPDATE_SERVICES to close the service on the source schema.
16. Sets the status flag of the source parent service to PIN_STATUS_FLAG_DUE_TO_SERVICE_TRANSFER and sets the status to **Closed**.
17. Sets the status flag of the source member services with PIN_STATUS_FLAG_DUE_TO_SUBSCRIPTION_SERVICE and sets the status to **Closed**.
18. Generates the **/event/customer/status** event.
19. Creates the **/event/audit/subscription/transfer** audit event to record the service transfer. The **/event/audit/subscription/transfer** event contains PIN_FLD_TO_SERVICE_OBJ, PIN_FLD_TO_BAL_GRP_OBJ, and PIN_FLD_TO_PROFILE_OBJ.
20. Calls the PCM_OP_SUBSCRIPTION_POL_POST_TRANSFER_SUBSCRIPTION policy opcode if the **TransferScheduledActions** business parameter is enabled.

If the source and target balance groups of the services transferred are in different schemas, PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER does the following:

1. Calls PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION, which finds all the services and their new service references and uses the references to purchase deals on the services in PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER.

   **Note:** For PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER to call PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION:

   - Either PIN_FLD_FROM_BAL_GRP_OBJ or the **/service** object must be present in the input flist.
   - Either the **/billinfo** object or the **/billinfo** array must be present in the input flist.

2. Creates the **/event/audit/service_balgrp_transfer** audit event to record the balance group transfer. The **/event/audit/service_balgrp_transfer** audit event contains PIN_FLD_TO_SERVICE_OBJ.

# 15

# Working with Profile Sharing Groups

This chapter describes how to create and use profile sharing groups in your Oracle Communications Billing and Revenue Management (BRM) system.

To use this documentation, you should be familiar with extended rating attributes (ERAs). See "About Extended Rating Attributes" in *BRM Configuring and Running Billing*.

## About Profile Sharing Groups

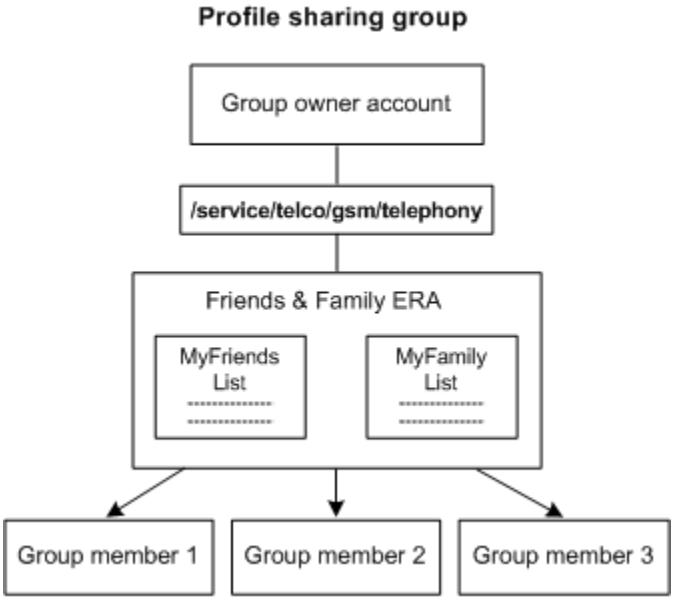
A profile sharing group enables an account or an account's service to share a profile with other accounts or services. A profile stores ERAs or other types of information about an account.

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.

**Important:** Only service-level profiles can be shared. Account-level profiles cannot be shared, even if the profile sharing group is owned by an account.

Profile sharing can be used to share an ERA, such as a friends and family list, to make it available to 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, as shown in Figure 15–1:

*Figure 15–1 Profile Sharing Group*




With this feature, you do not need to set up the same friends and family list or other ERA values separately for different accounts: they're shared automatically.

Profile sharing groups can also be used to share custom profiles you create.

You can set up profile sharing groups in two ways:

- Using Customer Center.
- Using a customized third-party client application. See "Creating a Profile Sharing Group through a Customized Client Application".

For guidelines that apply to both creation methods, see "Creating Profile Sharing Groups".

> **Note:** Profile sharing groups work similarly to charge and discount sharing groups, but profile sharing groups do not share resources and are not prioritized.

## About Profile Sharing Group Membership

Only service-level 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 forth, do not.

- **Service instance**: When you specify a service instance (for example, 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 phone service and personal phone service, a friends and family ERA might apply only to the personal phone.

Profile sharing group members should be equivalent to the owner. If the owner is a service, then the members should be the same service type. If the owner is an account, then the members should also be accounts.

Members do not need to have the same products or discounts as the owners. But to make use of a shared profile, members need a product or discount configured to use an ERA type or ERA label in the shared profile to give special rates or discounts.

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".

### How Account Status Changes Affect Profile Sharing Groups

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.

### How Group Owner Changes Affect Profile Sharing Groups

If you use a third-party client application, you can customize the application to allow a customer service representative (CSR) to change the owner of a profile sharing group. For more information on how to implement this, see "Changing the Owner of a Profile Sharing Group through a Customized Client Application".

> **Note:** Customer Center does not support changing group owners.

When the owner of a profile sharing group changes, the profiles shared by the previous owner are deleted from the group and the profiles shared by the new owner are added.

## 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 will be shared by the members.

You create profile sharing groups as follows:

- Using Customer Center.

- Using a third-party client application, see "Creating a Profile Sharing Group through a Customized Client Application".

The following guidelines apply when creating a profile sharing group:

- 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 forth do not.

## Creating a Profile Sharing Group through a Customized Client Application

For general information about profile sharing groups, see "About Profile Sharing Groups".

To customize a third-party application to create a profile sharing group, use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE opcode.

The profile sharing group has either a service or an account as the owner and a list of member services or accounts. It also contains one or more **/profile** objects owned by the group owner that will be shared with group members.

This opcode creates the group object and then adds the members and profiles:

1. Creates a **/group/sharing/profiles** object and assigns the group owner.
2. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/profiles** object. Each PIN_FLD_MEMBERS array element specifies an **/account** object and a **/service** object. If the **/service** object is NULL, the **/account** object is the member.

   If the PIN_FLD_SERVICE_OBJ field for a member specifies a type-only service, such as GSM, instead of a specific service instance, all instances of that service type become members of the profile sharing group. However, subclass instances of the service type do not become members.

   If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/profiles** object is created. The member account can join the group and purchase the service later. See "About Profile Sharing Group Membership".
3. Validates that the group owner does not belong to a profile sharing group owned by one of the members.
4. Calls the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode to validate the members. By default, this opcode has no validation rules for profile sharing group members, but you can customize it to validate members. See "Validating Profile Sharing Group Members".

**Note:** PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE does not validate that the primary currency of members matches the parent.

5. Adds the **/profile** objects in the PIN_FLD_PROFILES array to the PROFILES array of the **/group/sharing/profiles** object.

   The profiles must be owned by the profile sharing group owner and must have a current validity period.

   If the PIN_FLD_PROFILES array is empty, the profiles list will be empty. Duplicate entries are ignored.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE returns the Portal object ID (POID) of the profile sharing group object created and the POID of the event generated to record the creation.

When the profile sharing group is created, the group is automatically added to each member's ordered balance group. See "Adding a Profile Group to a Member's Ordered Balance Group".

PCM_OP_SUBSCRIPTION_SHARING_GROUP_CREATE fails if the owner is a member of a sharing group owned by one of the members.

### 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 discount and charge sharing groups, **/ordered_balgrp** controls the order in which the group's resource balances are impacted by events. See "How Discounts and Charges Are Applied" in *BRM Managing Accounts Receivable*.
- For profile sharing groups and balance monitor groups, the order is not significant, so they are added to the end of the list in the PIN_FLD_ORDERED_BALGROUPS array. The balance monitor groups are listed before the profile sharing groups.

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.

### Validating Profile Sharing Group Members

To validate the members of a profile sharing group, customize and call the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode. By default, this opcode validates only members of monitor groups.

You can customize this policy opcode to implement your validation rules. For example, the opcode can make sure that a member's service type is the same as the owner's service type.

If customized for profile sharing groups, this opcode validates members before they are added or modified and returns a list of valid members to the calling opcode.

## Modifying Profile Sharing Groups

You can modify profile sharing groups through Customer Center or by customizing a third-party application. You can modify a profile sharing group in the following ways:

- **Add members to the group**: When a member is added to the group, shared profiles become available to events generated by that member.

- **Add shared profiles**: When profiles are added to a profile sharing group, the profiles automatically are available to group members. The added profiles must be owned by the group owner and have valid dates.
- **Delete members from the group**: When a member is deleted from the group, 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 of a profile sharing group changes, members use the new owner's shared profiles instead of the old owner's shared profiles.

> **Note:** You cannot change the owner of a profile sharing group in Customer Center.

To modify profile sharing groups by customizing a third-party application, see the following:

- Modifying a Profile Sharing Group through a Customized Client Application
- Deleting Members and Profiles from a Profile Sharing Group through a Customized Client Application
- Changing the Owner of a Profile Sharing Group through a Customized Client Application

## Modifying a Profile Sharing Group through a Customized Client Application

To modify a profile sharing group, use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY opcode. This opcode performs one of the following modifications, depending on the input POID:

- Adds member accounts or services to an existing group.
- Adds profiles to an existing group.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY returns the POID of the group that was modified and the POIDs of the events that were generated to record the group modification.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY fails in the following cases:

- If the owner is a member of its own group or a group owned by a group member.
- If a profile object in the input flist is not valid.

For more information, see:

- Adding Members to a Profile Sharing Group through a Customized Client Application
- Adding Profiles to a Profile Sharing Group through a Customized Client Application

### Adding Members to a Profile Sharing Group through a Customized Client Application

To add members to a profile sharing group, use PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY.

**Important:** The element ID for each member in the input flist must be unique within the membership. If an element ID is already being used by an existing member in the group object you are modifying, the opcode overwrites the existing member.

To make sure that you do not assign a new member to an existing member's ID, include all the existing members in the input flist. In this case, the opcode will make sure that all new members are added, even if a new member's element ID is the same as an existing member's ID.

This opcode handles duplicate members, so including existing members in the input flist does not cause problems.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/account** and **/service** objects in the PIN_FLD_MEMBERS array to the MEMBERS array of the **/group/sharing/profiles** object.

   Each PIN_FLD_MEMBERS array element specifies an **/account** object and a **/service** object. If the **/service** object is NULL, the **/account** object is the member.

   **Note:**

   - The guidelines for creating a profile sharing group also apply to adding members to an existing group. See "Creating Profile Sharing Groups".
   - If you specify a service type as a member, the member account does not need to own the service when the **/group/sharing/profiles** object is modified. The member account can join the group and purchase the service later.

2. Validates that the group owner does not belong to a profile sharing group owned by one of the members.
3. Calls the PCM_OP_SUBSCRIPTION_POL_PREP_MEMBERS policy opcode to validate the members. By default, this opcode has no validation rules for profile sharing group members, but you can customize it to validate members. See "Validating Profile Sharing Group Members".
4. Creates an flist that includes the list of members to be added to the sharing group.
5. Generates an **/event/group/sharing/profiles/modify** event to record the changes.

Adding a member to a profile sharing group triggers the PCM_OP_SUBSCRIPTION_POL_AUTO_SUBSCRIBE_MEMBERS policy opcode, which adds the profile sharing group to the ordered balance group for each member. See "Adding a Profile Group to a Member's Ordered Balance Group".

### Adding Profiles to a Profile Sharing Group through a Customized Client Application

When adding profiles to a profile sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY does the following:

1. Adds the **/profile** objects in the PIN_FLD_PROFILES array to the PROFILES array of the **/group/sharing/profiles** object.
2. Validates that the new group name is not a duplicate of an existing group name.

3. Creates an flist that includes the list of profiles to be added to the sharing group.
4. Generates an **/event/group/sharing/profiles/modify** event to record the changes.

## Deleting Members and Profiles from a Profile Sharing Group through a Customized Client Application

> **Caution:** The element ID for each member in the **/group/sharing/profiles** object is unique. If you use an incorrect element ID for the member you want to delete, the opcode deletes the wrong member.

To delete a member or profile from a profile sharing group, your application calls PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY to list the members or profiles to be deleted. This opcode passes an empty array with an element ID for each listed object to be deleted.

> **Note:** If the array passed by the opcode is not empty, you are either adding or modifying the member or profile.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_MODIFY then calls the PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE opcode.

To delete a profile sharing group member, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. Calls the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode to delete the **group/sharing/profiles** object from the member's **/ordered_balgrp** object.

   For more information, see "Adding a Profile Group to a Member's Ordered Balance Group".
2. Deletes the member from the **/group/sharing/profiles** members array.
3. Generates an **/event/group/sharing/profiles/delete** event.

To delete a shared profile from the profile sharing group, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. Deletes the **/profile** objects specified in the PIN_FLD_PROFILES array from the PROFILES array of the **/group/sharing/profiles** object.
2. Generates an **/event/group/sharing/profiles/delete** event to record the deletion.

If successful, PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE returns the POID of the profile sharing group object that was modified and the POID of the event that was generated.

## Changing the Owner of a Profile Sharing Group through a Customized Client Application

To change the owner of an existing profile sharing group, use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT opcode. You cannot change the group owner in Customer Center.

This opcode takes the following fields as input:

- PIN_FLD_GROUP_OBJ: The profile sharing group (**/group/sharing/profiles**) whose owner is being changed.
- PIN_FLD_POID: The **/account** object of the current owner.
- PIN_FLD_PARENT: The new owner. This field identifies the **/service** object designated as the new owner of the profile sharing group. The **/service** object can be in the account that currently owns the profile sharing group or it can be in a different account.
- PIN_FLD_ACCOUNT_OBJ: The **/account** object of the new owner.

  **Note:** If you are changing the ownership of the profile sharing group from one service to another within the same account, the object passed in this field matches the one in the PIN_FLD_POID field.

- PIN_FLD_BAL_GRP_OBJ: The balance group used to track sharing activities for the new owner.
- PIN_FLD_PROFILES: An array of profiles that the new owner will share with the group members. This field is optional but should be included if the shared profiles for the new owning service are different than those for the current owning service.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_SET_PARENT does the following:

1. Verifies that:
   - The input flist contains the **/balance_group** object and **/account** object for the new profile sharing group owner.
   - The **/balance_group** object belongs to the new owning account or service.
   - There are no cyclical relationships. For example, the owner cannot be a member of its own group. Also, if a member of the group owns a profile sharing group, the owner cannot be a member of that group.
2. Deletes the current owner's profiles from the **/group/sharing/profiles** object.
3. Adds the list of profile instances of the new parent passed in the input flist to the group object.
4. Sets the **/group/sharing/profile** object's owner to the new owner specified in the PIN_FLD_PARENT input field.
5. Adds the new owner's shared profiles to the **/group/sharing/profiles** object.
6. Generates an **/event/group/sharing/profiles/modify** event to record the owner change.

If successful, this opcode returns the POID of the profile sharing group that was modified and the event that was generated to record the owner change.

This opcode fails if the new owner that is passed is already a member of the profile sharing group.

## Deleting Profile Sharing Groups

A profile sharing group is deleted when the group owner's account is closed or the sharing group is deleted by a CSR. When a profile sharing group is deleted, members can no longer use the profiles that had been shared by the owner.

When a profile sharing group is deleted, BRM also deletes the profile sharing group from each member's ordered balance group. For more information on ordered balance groups, see "Adding a Profile Group to a Member's Ordered Balance Group".

You delete profile sharing groups as follows:

- Using Customer Center.
- To delete a profile sharing group through a third-party client application, see "Deleting a Profile Sharing Group through a Customized Client Application".

To delete members or profiles from a group, see "Modifying Profile Sharing Groups".

## Deleting a Profile Sharing Group through a Customized Client Application

To delete a profile sharing group, use the PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE opcode.

If the member and profiles arrays are empty in the input flist, the opcode assumes you are deleting the group.

PCM_OP_SUBSCRIPTION_SHARING_GROUP_DELETE does the following:

1. Calls the PCM_OP_SUBSCRIPTION_ORDERED_BALGRP opcode to delete the **group/sharing/profiles** object from each member's **/ordered_balgrp** object.
2. Deletes the **group/sharing/profiles** object.
3. Generates an **/event/group/sharing/profiles/delete** event to record the deletion.

# 16

# Sending Email to Customers Automatically

This chapter describes how to set up, configure, and start the Oracle Communications Billing and Revenue Management (BRM) Email Data Manager (DM), **dm_email**. The Email DM is a process that enables you to send customer notifications and invoices automatically via email.

> **Important:** The Email DM is not related to the Email Manager, the optional BRM component that integrates your company's email service with the BRM system. You use the Email Manager to manage your customers' use of your email service. See "About Email Manager" in *BRM Email Manager*.

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

For more information on emailing invoices, see "Sending Invoices to Customers" in *BRM Designing and Generating Invoices*.

You also run Email DM to send welcome messages to new customers. For more information, see "Sending Welcome Messages to Customers".

## Running the Email Data Manager

This section describes how to run Email Data Manager.

### Starting the Email Data Manager

To run the Email DM:

1. Edit the configuration file for **dm_email**. See "Configuring the Email Data Manager".
2. Start the **dm_email** process:

   ```
   pin_ctl start dm_email
   ```

   This script starts the **dm_email** process.

For more information on how to specify the sender of automatic email messages, see "Specifying the Sender of Welcome Messages".

### Configuring sendmail Options

The Email DM (**dm_email**) uses the **sendmail** program.

By default, the DM uses the following **sendmail** command:

```
/usr/lib/sendmail -t
```

This default should be sufficient for most users. But you can configure BRM to use different command-line options for **sendmail**.

> **Caution:** Do not change the **sendmail** configuration unless you are an experienced **sendmail** user.

1. Open the Email DM configuration file (*BRM_Home*/**sys/dm_email/pin.conf**). *BRM_Home* is the directory in which you installed the BRM software.
2. Edit the **unix_sendmail_command** entry. You can specify different options for sendmail, but you cannot substitute a different program for sendmail.
3. Save the file.
4. Stop and restart the Email DM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

# Configuring the Email Data Manager

This topic covers configuring the Email DM (**dm_email**). Carry out these tasks:

- Editing the Email Data Manager Configuration File
- Checking Entries in the Connection Manager Configuration File

For more information, see the description of each task.

## Editing the Email Data Manager Configuration File

To edit the Email Data Manager configuration file:

1. Open the Email DM configuration file (*BRM_Home*/**sys/dm_email/pin.conf**).
2. Edit the file according to the instructions it contains.
3. Save the file.
4. Stop and restart the Email DM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

## Checking Entries in the Connection Manager Configuration File

Make sure you specify these required entries in the Connection Manager (CM) configuration file:

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. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

### Specifying the Sender of Welcome Messages

You have the option of specifying the sender of welcome email messages to new customers. You do this by editing the CM configuration file. See "Changing the Welcome Message Sender Address".

For other types of automatic email, the sender is always the default user who starts the **dm_email** process. For example, if the user **pin** runs **pin_ctl start dm_email**, the sender of your email messages is pin@yourcompany.com. The **pin** user is the user created to install BRM.

## Using the Email Data Manager to Send Printed Invoices

You can configure the Email DM (**dm_email**) to send invoices to a printer instead of emailing them. For more information, see "Configuring the Email Data Manager".

# 17

# Managing System and Account Currencies

This chapter describes how to set up the Oracle Communications Billing and Revenue Management (BRM) system currencies and account currencies.

## About System and Account Currencies

You specify two types of currency in your BRM system:

- System currency
- Account currency

The *system currency* is 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).

The *account currency* is 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.

To set the system currency, see "Setting the System Currency".

The account currency is set by the customer or customer service representative (CSR) when the account is created. After the account is created, you cannot change the account currency for an account. See "Defining the Currency for Individual Accounts".

To set the default account currency, see "Setting the Default Account Currency".

## About Currency Conversion

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 when 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.

When currency conversion rates change, you need to update the **/config/currency/conversionrates** object. See "Changing Currency Conversion Rates".

## 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.

> **Caution:** 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 sets the account currency for an account when the currency is not specified at registration.

1. Open the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**). *BRM_Home* is the directory in which you installed the BRM software.
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.

## Defining the Currency for Individual Accounts

You define account currencies when accounts are created. For more information on creating accounts, see Customer Center Help.

> **Important:**
>
> - 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.

### Currencies for Account Groups

A subordinate bill unit or sponsored account must use the same account currency as the parent or sponsor account. See "About Account Groups" in *BRM Managing Accounts Receivable*.

### Managing Currencies in Customer Accounts

You can do the following tasks when managing customers:

- Specify the currency when you create the account.

  You specify the account currency during registration by choosing from a list of currencies and secondary currencies.

  If your account uses two currencies, and the primary currency is the euro, the list of secondary currencies is limited to the EMU currencies for countries that are still in the currency crossover period. (Countries that joined before February, 2002 can only use the euro.)

  - To add or remove a currency resource, see "Supporting a New Currency".
  - To change the list of secondary currencies, see "Changing Supported Secondary Account Currencies".

- Display the currency in an account in either the primary or secondary currency. Some amounts are always displayed in the primary currency.

### Currency Display in the Event Browser

The Event Browser always displays the balance impact of an event in the account's primary currency. If the balance impact affects the system currency or secondary currency, the Event Browser shows the balance impact of the event in that currency also.

## Supporting a New Currency

To add a currency resource, use the Resource Editor in Pricing Center.

**Note:** If you add a resource when Pricing Center is running, you must refresh the price list snapshot to see the new resources.

For more information, see "About Resources" in *BRM Setting Up Pricing and Rating*.

### Finding Currency Codes

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

**Tip:** The ISO currency codes are listed in the *BRM_Home*/**include/pin_currency.h** file.

## Using Multiple Currencies in Your Price List

When you create your price list, you specify the currency resource for rate plans. Most rate plans use the system currency.

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

- Allow BRM to translate all currencies to the account currency.
- Create products in each currency.

## Creating Products in Multiple Currencies

Converting currencies can result in fluctuating charges based on fluctuating exchange rates. Table 17–1 shows how exchange rates can cause variations in how much a customer pays for the same event, even if the amount charged by the product stays the same:

***Table 17–1 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 products in multiple currencies.

For example, to charge for Internet access time, you could create products that include Internet usage rates 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 register, they see prices in a familiar currency.

**Tip:** To create multiple products by using different resources, create a set of products for one currency, then use the Product Creation wizard to copy the products for each currency.

## About Rating with Multiple Currencies

When rating an event, BRM tries to use a rate plan that charges in the account currency by default. If BRM can't find a rate plan that charges in the account currency, BRM searches for one that uses the system currency. If it can't find one that uses the system currency, it continues to the next product in priority order to rate the event.

# About Collecting Credit Card Payments in Multiple Currencies

When you use a credit card processing service, you must deposit payments in a different bank account for each currency. To do so, use a separate merchant ID for each currency, as explained in "About merchant numbers and account identifiers" in *BRM Configuring and Collecting Payments*.

**Note:** Credit card processors typically support only a small set of currencies. See the documentation for your credit card processor.

## 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 1, 2009 to July 31, 2009 and the period from August 1, 2009 to October 31st, 2009.

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.

**Important:** 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.

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

**Important:** Stop and restart the CM after editing the object.

## 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 register, they can select either the euro or the EMU as their primary account currency.

- When the primary account currency is an EMU currency, the secondary currency must be the euro. 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.

**Tip:** You can specify whether BRM requires a secondary currency when the euro is the primary currency. See "Changing Supported Secondary Account Currencies".

## 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, then you should make sure that customers who have an account currency different from the system currency cannot purchase products by using the system currency. To do so, use the euro as the primary currency for those accounts.
- Restrict all rates 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.

**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.

### Setting the Error Tolerance

Use the Resource Editor in Pricing Center to set error tolerance.

> **Important:** After you set error tolerance values for your supported currencies, you must activate the error tolerance by modifying the Connection Manager (CM) configuration file. See "Rounding Errors for Overpayments and Underpayments".

### 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 from the Resource Editor.

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 in *BRM_Home***/sys/cm**.

## Limiting the Number of EMU Currencies

You can limit the EMU currency choices available for account currencies. For example, you might want to offer only the currencies in the countries where you do business. See "Changing Supported Secondary Account Currencies".

## Euro Support in BRM Client Tools

BRM client applications support multiple currencies in the following ways:

- Use Event Browser to display the euro and EMU currencies and their balance impacts.

> **Note:** If the event involved the primary currency, you see only the primary currency. If the event involved the secondary currency, you see the primary and secondary currencies.

- Use Customer Center to toggle views between the euro and EMU currencies.
- Use Payment Tool to enter payments in the euro or EMU currencies.

## Assigning Primary and Secondary Account Currencies at Registration

When you create an account in Customer Center and select an EMU national currency, the euro is specified as the secondary currency automatically. If you select the euro as

the account currency, you must choose an EMU national currency for the secondary currency.

**Tip:** You can configure BRM to not require a secondary currency when the euro is the primary currency. See "Changing Supported Secondary Account Currencies".

## Changing the Euro Conversion Error Tolerance

You set the euro conversion error tolerance in the Resource Editor, an application in the Pricing Center. See the Resource Editor help.

## Changing Supported Secondary Account Currencies

To limit the EMU currency choices displayed in Customer Center, you need to 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
```

**Important:** Stop and restart the CM after editing the object.

**Tip:** 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 registration, the default secondary currency is the first supported currency listed.

**Tip:** 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 II

## Implementing Branded Services

Part II describes how to implement branded services. It contains the following chapters:

# 18

# About Branding

This chapter provides an overview of brands and describes brand-aware functionality in a single Oracle Communications Billing and Revenue Management (BRM) system.

You must have the Brand Manager component to create and use brands.

## About Managing Multiple Brands of Service

You can use BRM to host Internet services for branded service providers (BSPs). A BSP is any business that offers Internet access and other services under its own brand identity without building the infrastructure necessary to support the services. Instead, a BSP retains its own brand image and relies on a host service provider (HSP) to provide services to its subscribers.

For example, if a credit card company wants to provide Internet access to its customers under the credit card brand name, it can have an HSP supply the service. The HSP uses the credit card logo and brand image on all interfaces to the subscriber, including billing invoices and self-administration Web sites. The service subscribers never know that the Internet service they ordered from the credit card company is provided and managed by a different Internet service provider (ISP). Any corporation can use its name and customer base to offer Internet services.

You can also use the BRM branding functionality to divide one company into distinct groups, such as regional sales divisions. The only difference is that the HSP is not concealed; it controls all services and billing under its own logo and brand image because the brands are part of the same company. Each brand has its own pricing structure, general ledger (G/L) accounts, and control of the brand-specific interfaces to the BRM database. The HSP sets up and maintains each brand in a single BRM system.

An HSP can:

- Create a brand, by using the BRM mechanism for keeping customer accounts and other data for a BSP separate from data for other brands in the same BRM system. See "Creating Brands".
- Control which BRM personnel can access customer accounts and other data in each brand. See "Controlling Access to Top-Level Brands".
- Create separate price lists for each brand. See "Creating a Price List for Brand Accounts".
- Create brand-specific Web pages for customers to register and modify their accounts. See "Creating a Brand-Specific Web Page".

You need the Brand Manager component to host services for BSPs. See "When to Use Account Groups Instead of Brands" if you are not certain if your company requires branding.

# Understanding Brands

A brand is implemented as a type of BRM account. You create and modify brands by using the Brand Manager component of Configuration Center.

**Note:** Because brands are a type of account, you can also view and modify them in Customer Center.

When you create a brand, you include the login names of one or more brand administrators who manage that brand. Only the host administrator, a brand administrator, or a customer service representative (CSR) with access privileges for the brand can create and modify the brand-specific data. Use Access Privileges, another Configuration Center application, to give additional BRM users access to a brand. For more information, see "About Granting Access to Brands".

After the host administrator creates a brand, the brand administrator creates the price lists, customer accounts, and other data that is specific to that brand.

## About Brand-Aware Data

When using certain BRM client applications such as Customer Center and Configuration Center, you select a brand before you create or modify any data. You must either be a brand administrator or have access privileges for a brand to select that brand in an application. (This is not a requirement for Customer Center; you can open accounts from multiple brands at the same time.) See "About Granting Access to Brands" for more information.

When you select a brand, the BRM system limits your access and allows you to only modify the data saved in that brand; for example, pricing information and general ledger information. This data can be configured uniquely for each brand in one BRM system.

Some BRM data is system-wide and is shared among all brands. For example, any brand administrator with proper permissions can read the data dictionary, which contains all BRM storable classes. Some data in the data dictionary, such as resources, impact categories, and usage maps, cannot be configured specifically for one brand because the data is shared among all brands. If you do not want one brand to draw information on the strategies of another brand based on shared data, the host administrator should name configuration data cryptically or non-descriptively. See "About Duplicate Service Logins Across Brands" for an example of how to name services.

### Brand-Aware Information

Table 18–1 describes how branding affects certain BRM applications and features. The descriptions in the table apply only when you implement brands.

***Table 18–1 Branded Data Behavior***

| Branded Data | Branding Behavior |
|---|---|
| Access Privileges tool | You use the Access Privileges tool to add users to an access group for a brand or account group. The access group limits who can access and manage the branded accounts. See "About the Root Access Group". |
| Account sponsorship | Sponsorship is valid within a brand; an account in one brand cannot not sponsor an account in a different brand, including sub-brands. |

*Table 18–1 (Cont.) Branded Data Behavior*

| Branded Data | Branding Behavior |
|---|---|
| Customer Center | You create and change accounts only in the brand you select from the **Brand** list in Customer Center. Only brand administrators and CSRs with proper access privileges can see brand-specific information and accounts. |
| Brand Manager | Host administrators and brand administrators log into Configuration Center and use Brand Manager to create brands and sub-brands. See "Creating Brands". |
| Credit card merchant IDs | All brands must use the credit card processing service configured for the BRM system; however, you define merchant IDs for each brand. See "Setting Up Payment Methods for Brands". |
| Customer Center | Only brand administrators and CSRs with proper access privileges can see brand-specific information and accounts. |
| Device Management framework | **/device** objects and the **/config** objects used to configure them are brand-aware. You can associate **/device** objects only with **/service** objects in the same brand. You can specify different device life cycles and service mappings for device types in different brands. See "Device Management and Brands" in *BRM Developer's Guide* for more information. |
| Event Browser | The events available through Event Browser are those related to the login and password that the CSR uses to log in to Customer Center. You can review these events to see the changes a CSR makes to accounts in a brand the CSR has access to. |
| G/L IDs | The host administrator defines one set of G/L IDs for the database and one G/L segment for each brand in the database.<br>You set up G/L information for brands in the **pin_glid** file. |
| BRM Reports | The host administrator must install and configure BRM Reports, after which brand administrators can create reports for their brand. See "Reporting by Brand" in *BRM Reports* for more information. |
| Invoice Browser | The host administrator can change the default invoice template for a BRM system.<br>Brand administrators can design custom invoice templates and save them to their brands. |
| Multischema support | If your BRM system is installed in a multischema environment, you can install each top-level brand in a chosen database schema. All accounts and sub-brands in a brand must reside in the same schema as the brand account. |
| Payment Tool | To process payments in a branded environment, you must be logged in to Payment Tool as the **root** user. After you are logged in, you can only create and change payment batches in the brand you select from the **Active Brand** list.<br>See Payment Tool for more information. |

*Table 18–1 (Cont.) Branded Data Behavior*

| Branded Data | Branding Behavior |
|---|---|
| Pricing Center | Pricing Center is brand-aware based on the login of the brand administrator; the price list displayed in the tool is for that brand. You cannot switch between brands after you log in to Pricing Center.<br>All pricing components, including products, deals, plans, plan lists, and price lists, are brand aware; brands cannot share pricing components.<br>Only the brand administrator for a brand can create the price list for that brand. See "Creating a Price List for Brand Accounts". |
| Self-Care Manager registration | You can create a customer self-care home page for each brand in your system. Each self-care home page includes code that specifies the brand to use when displaying account information. See "About Registering Customers". |
| Zone Mapper | Host administrators and brand administrators log into Pricing Center and use Zone Mapper to create zone maps for pricing and save them to a specific brand. |

## System-Wide Information

Data that cannot be separated according to a brand should be handled only by the host administrator because all brands access and use the same data. System-wide applications and objects include the data shown in Table 18–2:

*Table 18–2 System-wide Data and Branding Behavior*

| System-wide Data | Branding Behavior |
|---|---|
| Billing utilities | The host administrator should run billing scripts, such as **pin_bill_day**, for all brands at one time. Billing for one brand is possible, but it is resource intensive. See "Running Utilities with a Branded Database". |
| Configuration information | Configuration information such as ratable usage metrics (RUMs), currencies, and tax supplier IDs are system-wide, with the exception of the following: **/config/glid**, **/config/invoice_templates**, and **/config/invoice_events**. These can be configured for a specific brand. See "Setting Up the General Ledger to Support Brands" and "Setting Up Invoicing to Support Brands". |
| Credit card processing | All brands must use the payment processor connected to the credit card data manager; for example, Paymentech. See "About BRM-Initiated Payment Processing" in *BRM Configuring and Collecting Payments* for more information. |
| Developer Workshop and the data dictionary | The BRM object schema is a shared resource between all brands; therefore, only the host administrator should modify it. |
| Facility Module (FM) policy opcodes | By default, FM policy opcodes are not brand-aware; however, some policy opcodes will pass brand-aware data. |
| Services | Although service names are system-wide and all brands can share the same service object, each brand uses a different instance of the service. |
| Taxation | All brands in a BRM system must use the same tax calculation software. |

*Table 18–2 (Cont.) System-wide Data and Branding Behavior*

| System-wide Data | Branding Behavior |
|---|---|
| Remittance for service providers | A brand account cannot contain a remittance product; however, you can set up a separate remittance account in a branded BRM system. The account receiving remittance can be in the same brand as the accounts paying the remittance funds. The remittance account can also be part of another brand or a system-level account. |

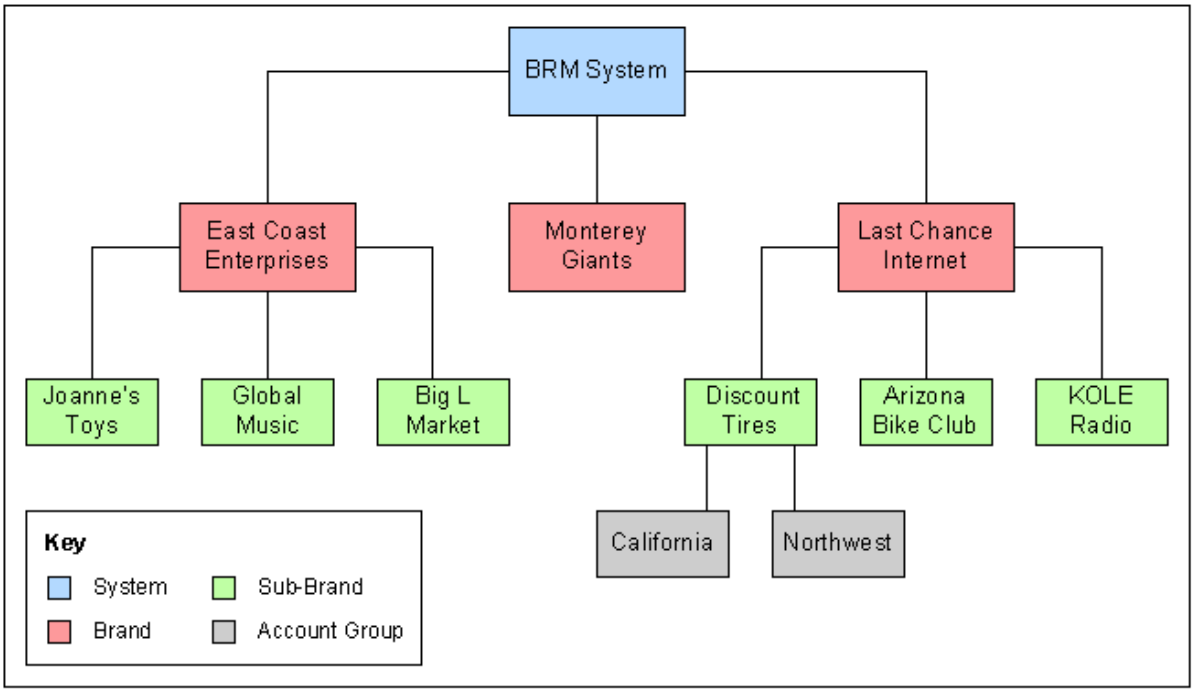
# About Organizing Brand Hierarchies

You organize accounts within a brand by using sub-brands and account groups:

- Create sub-brands in Brand Manager. For example, if a brand has regional offerings or subsidiary brands, you can create hierarchies of sub-brands within the parent brand.
- Create account groups in Customer Center. For example, you might want to assign different CSRs to different groups of accounts by using access privileges.

Figure 18–1 shows how a brand hierarchy might be organized, with a mixture of brands, sub-brands, and account groups:

*Figure 18–1 Brand Hierarchy*




## About Naming Brands

Under the same parent brand, each sub-brand must have a unique name. However, sub-brands under different parent brands can share the same name. For example, in the figure above, BRM cannot contain two top-level (parent) brands named East Coast Enterprises. However, the East Coast Enterprises brand and the Last Chance Internet brand can each contain a sub-brand named Global Music. This is valid because the

Global Music sub-brands are part of different brand hierarchies that have different parent brands.

**Note:** The same rules apply for sponsor groups. You cannot have two sponsor groups with the same name under one parent brand.

## About Brands and Account Groups

All accounts under one brand are in the same account group. One brand can contain any number of sub-brands and account groups that do not have to be brands. Account groups can be organized into groups for billing and reporting. See "About Account Groups" in *BRM System Administrator's Guide* for more information.

A brand can contain an unlimited number or type of account groups, but certain rules apply when moving accounts between brands and account groups to keep data separated.

- You cannot transfer a customer from one brand to another brand or to a non-branded mode.
- You cannot move an account from a non-branded to a branded mode.
- You can move an account from one account group to another account group within a brand, but not between brands.
- You can drag nonpaying (subordinate) child accounts and paying (nonsubordinate) child accounts between account groups within the same brand.

### When to Use Account Groups Instead of Brands

Before you create customer accounts in your BRM system, first determine if you require branding to separate accounts into divisions or if using account groups meets your requirements. You can create account groups whether or not you enable branding.

Use account groups rather than branding if the following circumstances are true:

- You plan on moving accounts in and out of different clusters. Moving an account from one brand to another, or from a non-branded environment to a brand, is prohibited by branding and requires the accounts to be closed and recreated, which is a time-consuming process.
- You will share BRM pricing data such as products, deals, and plans, or general ledger revenue reporting. This eliminates the need to copy the same pricing component into each brand's price list and then maintain multiple instances of that component.
- You need strict divisions between account hierarchies and want to segregate data into isolated groups.
- You will have sponsored accounts. You cannot apply sponsored rates across brand boundaries, and you cannot apply sponsored rates to a parent brand for accounts in a sub-brand.

See "About Brand-Aware Data" for more information.

## About Creating Sub-Brands

You use Brand Manager to create any number of sub-brands in a brand. Before you create a sub-brand:

- You must have access to the parent brand, either as a brand administrator or through access privileges. See "Controlling Access to Top-Level Brands".
- You must create a custom price list and commit it to the parent brand before you create sub-brands.
- Brand administrators must have **service/admin_client** permissions to log into Configuration Center and create brands. Therefore, when host administrators create brands in their BRM system, they must include **service/admin_client** in the default plan for the brand. This service gives brand administrators access to BRM client tools. Similarly, when creating sub-brands, brand administrators must include **service/admin_client** in the default plan for the sub-brand if they want sub-brand administrators to create additional sub-brands.

You can change most information for a brand, but you cannot change the plan or logins for brand managers.

## About Granting Access to Brands

You control which BRM users can access data belonging to a brand or an account group. Open the Access Privileges tool in Configuration Center and create an access group.

Two categories of BRM users have access to brands without belonging to access groups:

- The host administrator has root access to the entire BRM system.
- Brand administrators have full access to their brand and any of its sub-brands. Brand administrators are the login names entered as part of creating a brand. You can create one or more brand administrators when you create a brand, but you cannot change or delete them later.

Other BRM users, such as CSRs, must be part of a brand's access group to work with customer accounts and other data for that brand. For more information on creating access groups, see "Controlling Access to Top-Level Brands".

An access group includes:

- A single brand, sub-brand, or account group
- The logins of one or more BRM users

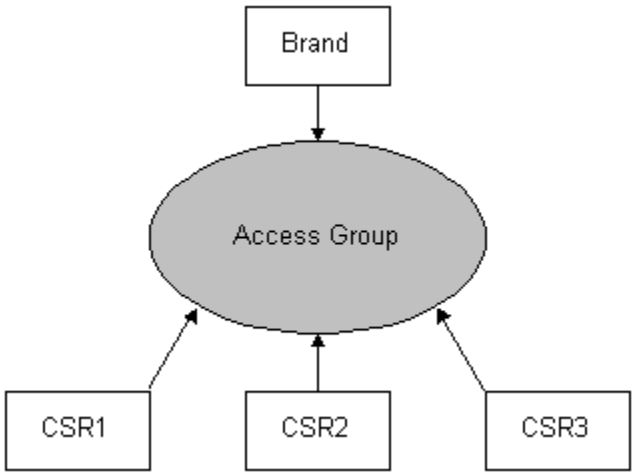
You can assign a BRM user to one or more access groups.

When using Customer Center or other BRM tools, you only see and work with data that belongs to the brand or account group you have access to. In Customer Center, you can access multiple brands concurrently if you have proper permission.

**Note:** Only the brand administrator can create a price list for a brand. Access group members do not have this ability.

Figure 18–2 shows the structure of an access group:

*Figure 18–2 Access Group Structure*




**Note:**

- If you have access to a brand, you also have access to its sub-brands. You do not have implicit access to parent brands in your brand hierarchy.
- Access privileges also apply to creating access groups. You need privileges to work with a brand to set up access groups for sub-brands or account groups within that brand.
- In Customer Center, CSRs belonging to multiple access groups can open accounts from multiple brands concurrently.
- You can add CSRs to an access group only if the CSRs already have access to a brand or account group within the hierarchy of an active brand. If the CSR was created in another brand hierarchy, the administrator of the parent brand must add the CSR to the desired brand.

Table 18–3 lists each access level available in a branded system.

*Table 18–3 Available Access Levels*

| This Access Level | Grants this Permission |
|---|---|
| Brand Account Administrator<br>*The user's login ID is used to create a brand.* | The brand administrator can access all accounts in the brand or account group and all sub-brands and sub-account groups. You do not need to add a brand administrator to an access group for that brand or account group. |
| Access Group Member<br>*The CSR's login ID is added to an access group for a brand or account group.* | The CSR can access all accounts in the brand or account group that the access group was created for, including sub-brands and sub-account groups in the hierarchy. |
| Brand Member<br>*The CSR account was created in a brand, but the CSR is not a member of an access group.* | The CSR can access only the accounts that he or she created. The access level in Customer Center is shown as [Self]. |

## About the Root Access Group

When you install BRM and enable branding, a default access group, or access control list (ACL), is created for the root database. The host administrator assigns a personal login ID as the owner, and then adds any number of CSRs to the list. Members of the root access group for BRM can see and change all accounts in the database.

For more information, see "Giving Users Access to All Accounts in BRM".

# About Price Plans for Brands

To create a brand, the host administrator must first create a host price list for the BRM system. This price list is never used for rating; it only allows for the creation of brands. The price list must contain a plan list named **brand**, which contains plans needed to create brands. A plan provides services to the brand that enable the brand administrator to perform necessary tasks. The following services are often required by brand accounts:

- **admin_client** service for any brand to enable access to BRM client applications.
- **pcm_client** service if you use applications, such as testnap, that log in to BRM through this service.

When you create a plan in Pricing Center, you can give the plan any name, but you must include it in a plan list named **brand**. Figure 18–3 shows a brand host price list that contains the **brand** plan list. The plan list contains a plan named *Brand Plan* that contains two services to create top-level brands with.

***Figure 18–3 Brand Host Price List***


After the BrandHost price list is committed to the BRM database, the plan *Brand Plan* will be displayed in Brand Manager.

**Note:** Unlike Customer Center, Pricing Center limits access to multiple brands. When logging in to Pricing Center, you can only see the price list for the brand whose login and password you used. Therefore, when the host administrator logs in to Pricing Center as **root**, the host administrator can only see the Brand Host plan, and not the price plans created by each brand. To see a brand price list, the host administrator must log in to Pricing Center by using the brand administrator's login and password.

## About Brand Currency

Brands and account groups can have a different currency from the system currency, and sub-brands and account groups can have a different currency from the parent brand or account group.

Brands and sub-brands can also be created in the system currency, even if the parent brand has a different primary or secondary currency.

For more information on account currency and system currencies, see "Managing System and Account Currencies".

## About Closing, Inactivating, and Reactivating Brands

You can set a brand to be active, inactive, or closed from either Brand Manager or Customer Center.

When inactivating or closing a brand, the following rules apply:

- All accounts you create in a brand are created with the same status as the brand.
- Inactivating a brand inactivates all accounts in the brand. It does not inactivate the sub-brands or their accounts.
- A brand and all of its accounts can be reactivated if no changes were made to the brand or accounts in the brand after the brand was inactivated.
- If you inactivate an account before inactivating its parent brand, the account will be reactivated when the brand is reactivated. If desired, you must manually deactivate it again by using Customer Center.
- Inactivating a brand moves all closed accounts in the brand to inactive status.

For more information, see "Changing Account and Service Status".

## About Duplicate Service Logins Across Brands

User access to most BRM objects is validated by using the object's Portal object ID (POID). However, user access to a service is validated by combining the service *name* with the user's login name and then checking the user's password information. Therefore, if two brands use the same service and define the same login name for that service, errors will occur.

To ensure that only one brand contains a particular login/service combination, you configure unique service subclasses for each brand. For example, Table 18–4 shows host ISP database containing two brands. Each brand uses the IP service, but each instance of the service is named differently for each brand:

***Table 18–4 Service and Login Names for Brands***

| Brand Name | Service Name | Login Name |
|---|---|---|
| A1 Internet, Inc. | **/service/ip/ISP_A** | **admin** |
| Best Internet, Inc. | **/service/ip/ISP_B** | **admin** |

Because the service names are different, the login name can be the same because the actual service is not shared between brands. But, if you use the same service *name* across brands, the login names must be unique across all brands that use that service name; it is not sufficient that the service instances be unique to a brand.

**Tip:** Use a generic naming convention for services to limit the transfer of company-specific information among brands. Notice that the services in the example are named **/service/ip/**ISP_* rather than the company name, such as **/service/ip/**A1.

See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide* for more information on creating service subclasses. If you do not want to duplicate services across brands, you can modify the login policy to append the brand's domain name to the login name when a customer registers. For more information, see "Adding or Changing Login Options" in *BRM Developer's Guide*.

## About Using Brands with Multiple Database Schemas

If your BRM system is installed in a multischema environment, you can choose a specific database schema when you create each top-level brand. All sub-brands and accounts that belong to a brand must be in the same schema as the parent brand.

Only the host administrator can specify a schema for a brand.

**Tip:** Before choosing a schema for a particular brand, take into account the size of each schema and the number of accounts you expect in each brand. You cannot move a brand or its accounts to another schema after you create them.

## About A/R Accounts and Brands

By default, a brand account is not designed to handle accounts receivable (A/R) operations. If a CSR creates a customer account and makes it a subordinate bill unit to the brand account, the services used by the customer might be received for free because the brand account is not set up for billing. Brand Manager does not enable you to configure billing information when you create the brand account; you must set this up in Customer Center after you create the brand account.

To prevent A/R problems, use Customer Center to create a separate business account in the brand and make it the designated A/R account.

**Note:** Although it is not recommended, you can configure the brand account to handle A/R. Open the brand account in Customer Center and set up the billing and payment information. Then, when you make a customer account nonpaying (subordinate) to the brand account, change the nonpaying child account's billing date to match the brand bill unit's billing date.

## About Running Billing

Using brands does not change the way you run billing for your BRM system. The host administrator should run billing for the entire database at one time.

However, you can run billing for each brand individually. It is not recommended because the process is very resource-intensive and slows system performance. See "Running Utilities with a Branded Database" for details.

# 19

# Configuring a Branded Database

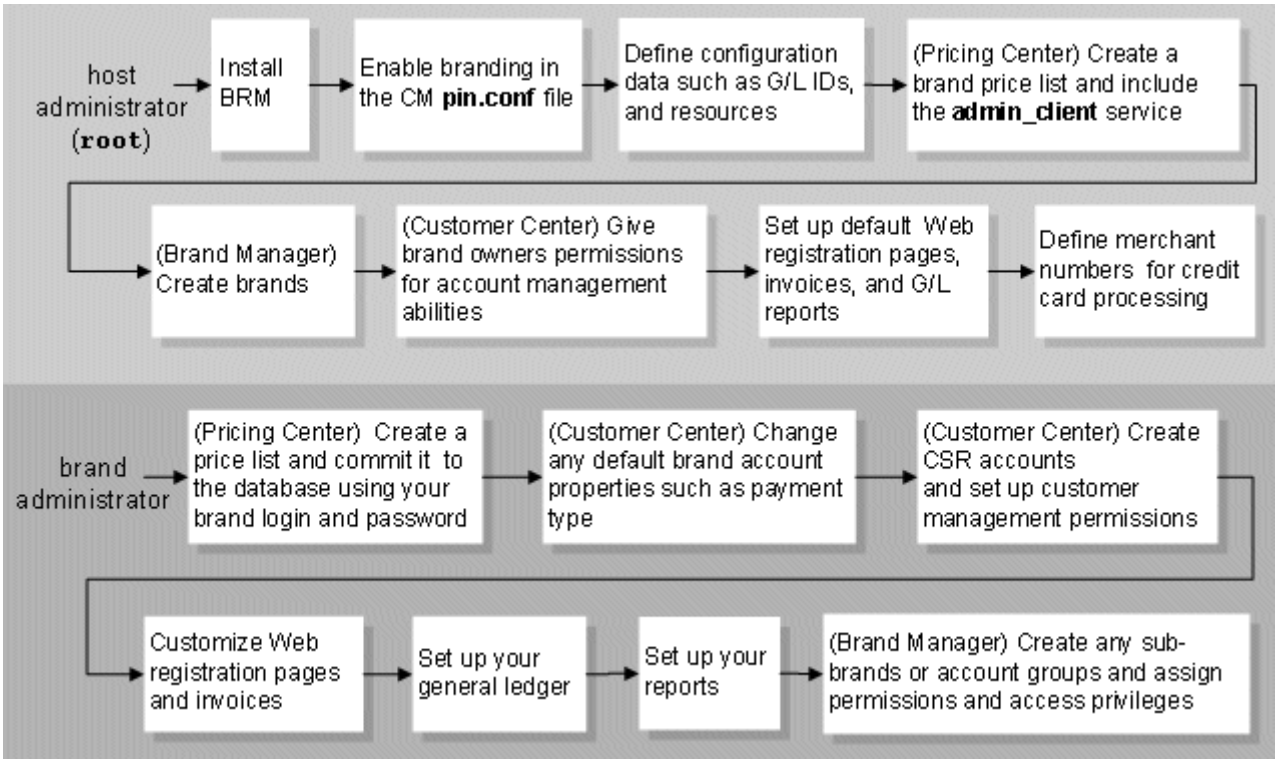
This chapter describes how to configure a Oracle Communications Billing and Revenue Management (BRM) database to use branding. The tasks described in this section are generally done by the host administrator.

You must have the Brand Manager component to create and use brands.

## Overview of Setting Up a Branded Database

Figure 19–1 shows the steps needed to create a brand and the person who typically performs each step.

***Figure 19–1 Brand Creation Steps***




Before creating a brand, you must set up your BRM system to use branding, and you must create a brand plan list.

To create a top-level brand, you must have root access to BRM. To create a sub-brand, you must have access to the parent brand, either as a brand administrator or through access privileges.

# Configuring BRM to Use Brands

The tasks described in this section are generally done by the host administrator before brands are created in the system.

## About the Host Administrator

The host administrator is the person with root account access to the BRM system. You need root account access to create a brand at the top-level of a brand hierarchy.

The host administrator sets up and administers the BRM database, which includes performing the following system-wide tasks and brand-related tasks. The system-wide tasks are common to all BRM databases; the host administrator must complete them whether branding is enabled or not. The brand-related tasks must be completed before branded service providers can manage brands.

### System-wide tasks

- Starting and stopping all system processes and services.
- Creating and changing BRM objects in the data dictionary. See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.
- Defining resources, ratable usage metrics (RUMs), and other pricing components. See "About Creating a Price List" in *BRM Setting Up Pricing and Rating*.
- Defining the default general ledger (G/L) ID account maps and G/L reports. See "Reporting by Brand" in *BRM Reports*.
- Setting up invoicing. See "Designing and Generating Invoices" in *BRM Designing and Generating Invoices*.
- Setting up Web registration for customers.
- Setting up and maintaining the terminal server. See "Understanding RADIUS Manager" in *BRM RADIUS Manager*.
- Running billing. See "Running Billing Utilities" in *BRM Revenue Assurance Center Online Help*.

### Brand-related tasks

- Activating Branded Service Management
- Defining a Price List for Brand Creation
- Controlling Access to the Data Dictionary
- Creating Brands
- Giving Permissions to the Brand Administrator
- Controlling Access to Top-Level Brands

## Activating Branded Service Management

You activate branded service management in your BRM system by editing the Connection Manager (CM) configuration file.

**Caution:** After you make branded service management active in your BRM system, *do not* try to change your system back to a non-brand system. You can corrupt data if you do this.

To activate branded service management:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**). *BRM_Home* is the directory in which you installed the BRM software.
2. Change the **dm_attributes** entry to include these parameters:

   ```
   - cm  dm_attributes database_number assign_account_obj, scoped, searchable
   ```

   If your BRM system uses multiple database schemas, you need a **dm_attributes** entry for each schema.

   See the configuration file for more information on this entry.
3. Change the **enforce_scoping** entry to activate branding:

   ```
   - cm  enforce_scoping  1
   ```

4. Save the file.
5. Stop and restart the CM.

## Defining a Price List for Brand Creation

The host administrator must first create a default host price list and save it to the BRM database before creating top-level brands. This price list does not contain products and rates; it only provides the services necessary for administrators to create brands. Brand administrators then create brand-specific price lists to define the rating for their brands.

**Note:** Only a brand administrator should commit a price list to BRM for a brand.

To create a default price list:

1. Log in to Pricing Center as **the root** user.
2. Define the plans and add services to them.

   **Important:** The plan you use to create brands must contain the **admin_client** service; without it, the brand administrator cannot use BRM client tools such as Configuration Center. If the brand administrator will use BRM developer applications such as **testnap**, also include the **pcm_client** service in the plan.

3. Create a plan list named **brand**; for the plan list type, click **new**.
4. Add the brand plans to the plan list.
5. Save the price list to the BRM database.
6. Create brands by using Brand Manager. See "Creating Brands".

For detailed instructions on creating a price list, see the Pricing Center Help.

**Tip:** To configure each brand with a default price list, commit the same price list to BRM every time, but use the brand administrator logins specific to each brand.

### Establishing Unique Service Logins

To ensure that only one brand contains a particular login/service combination, the host administrator can modify the customer login policy to extract the brand's domain name from the brand account's URL field and append it to the customer's login. The domain used for the customer's login is the domain defined for the brand in Configuration Center. For example, if the URL for the brand "East Coast Enterprises" is **www.ecoastent.com**, and a user has the login **jmcgee**, the login policy changes the login to **jmcgee@ecoastent.com**.

For an example of how to enforce unique email addresses for branded customers, see "Adding or Changing Login Options" in *BRM Developer's Guide*.

**Tip:** If you do not want to modify policy code, you can create a separate instance of a service for each brand. For more information, see "About Duplicate Service Logins Across Brands".

### Controlling Access to the Data Dictionary

The data dictionary that contains the object schema for the BRM system is not brand aware. Host administrators can restrict brand administrators from changing the data dictionary by disabling write access in the Data Manager configuration file. Access is disabled by default, but if host administrators make any changes, they must reset the access permissions to disable any future changes. See the DM configuration file for more information.

In addition to restricting access to the data dictionary, host administrators can use Developer Workshop to set **read**/**write** permissions on specific BRM storable classes. See Developer Workshop for more information on object properties.

For a non-branded system, updates to the data dictionary are also enabled and disabled using the DM configuration file.

## Creating Brands

**Note:** After you create a brand, you cannot change the brand plan.

For details on these steps, see the Brand Manager Help.

To create a brand:

1. Using Pricing Center, create a plan list named **brand** and commit it to the BRM database. See "Defining a Price List for Brand Creation".
2. Log in to Configuration Center as the **root** user.
3. Click the icon for Brand Manager shown in Figure 19–2:

*Figure 19–2 Brand Manager Icon*

4. Click **New Brand** and specify the following brand account information:
   - A unique brand name and the URL for the brand Web site.

     **Note:** Brand names must be unique within a brand hierarchy; each sub-brand under the same parent brand must have a different name. The brand URL is optional.

   - The currency used by the brand accounts.

     If you choose an EMU or the euro currency for the primary account currency, a **Secondary Currency** field displays. You must choose a second account currency for the brand.
   - (Optional) The brand status.
   - (Optional) The brand expiration date.
   - The brand plan.
   - The service login names and passwords for brand administrators.

   After you define the basic account information, you must add a billing contact to the brand.
5. Enter the following information on the **Contact** tab:

   **Note:** The billing contact information is required, but you can also define additional contacts. For more information, see the Brand Manager Help.

   - First and last name.
   - Email address.
   - Company name.
   - Mailing address.
   - (Optional) Phone number.

   **Tip:** Brands are identified in the Customer Center Account Navigator and in the **Search Results** lists by the account number and the billing contact's last name and first name, not by the brand name. To display accounts by brand name, enter the brand name in the billing contact's **Last Name** field.
6. Use Access Privileges, another Configuration Center application, to assign additional BRM users permission to work with a brand. For more information, see "Controlling Access to Top-Level Brands".

## Example of Using Brand Manager

To create a new brand called East Coast Enterprises:

1. Click the Brand Manager icon on the Configuration Center toolbar.

   Because this is the first brand created for this system, **Brand Host** (the BRM system) is the only item on the active brand list, and it is already selected.

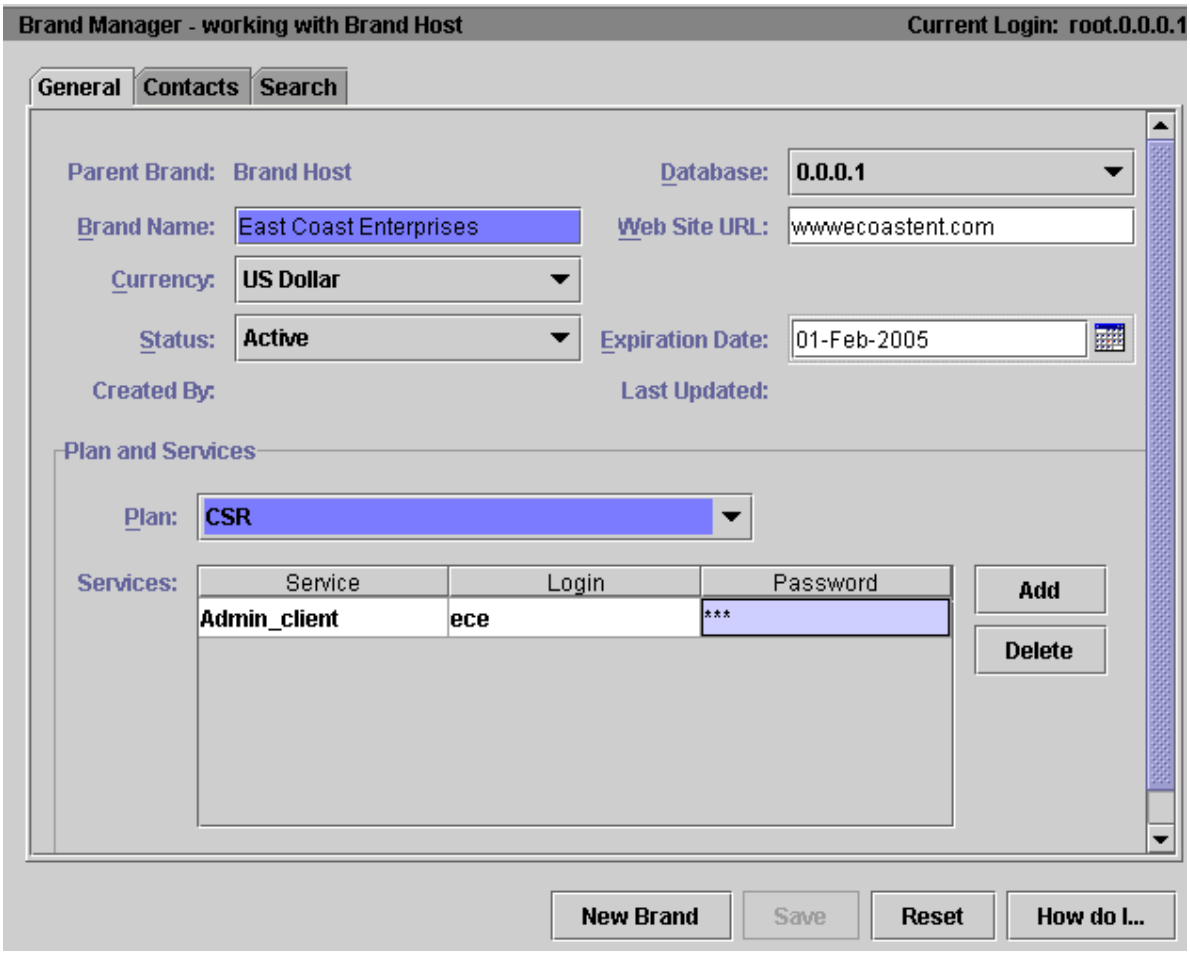
2. Enter the information on the **General** tab:

   - **Brand Name**: East Coast Enterprises
   - **Brand URL**: www.ecoastent.com
   - (In a multischema environment) **Database**: 0.0.0.1
   - **Currency**: US Dollar
   - **Status**: Active. By default, all accounts you create in the brand will be created with this status.
   - **Expiration Date:** February 1, 2005
   - **Plan**. The brand plan is **CSR** in this example
   - **Service**: admin_client
   - **Login**: ece
   - **Password**: ece

   **Note:** The brand administrator can change the login name and password at a later date by using Customer Center.

   Figure 19–3 shows the **General** tab for the host brand after you enter the data:

*Figure 19–3 Brand Manager General Tab*




**Note:** The **Save** button is inactive because billing contact information needs to be entered on the **Contacts** tab. You cannot save the brand until all required data is entered.

3. Click the **Contacts** tab and enter information for the billing contact.

**Note:**

- You can enter other contacts, but billing contact information is required.
- These contacts are only for your information and are not used for billing purposes.
- The phone numbers are optional.
- Brands appear in Customer Center as regular accounts but are identified in the Customer Center Account Navigator and in the **Search Results** lists by the account number and the billing contact's last name and first name, not by the brand name. To display brand accounts by brand name, enter the brand name in the billing contact's **Last Name** field.

4. Click **Save**.

# Setting Up Brands for Brand Administrators

The tasks in this section should be done by the host administrator before brand administrators customize brands and register customer accounts.

Preparing brands for brand administrators includes the following tasks:

- Controlling Access to Top-Level Brands
- Giving Permissions to the Brand Administrator
- Setting Up a Customer Center Search Filter for Access Group Owners
- Setting Up Invoicing to Support Brands
- Setting Up the General Ledger to Support Brands
- Setting Up Reports to Support Brands
- Setting Up Web Registration to Support Brands
- Setting Up Payment Methods for Brands
- Running Utilities with a Branded Database

## Controlling Access to Top-Level Brands

When using certain BRM applications, such as Payment Tool and Configuration Center, you must select a brand before you create or modify any data. You must either be a brand administrator or have access privileges for a brand to be able to select that brand in an application.

**Note:**

- This restriction is not true for Customer Center; you can open accounts from any brand that you have permission to access without having to select a specific brand.
- As a host administrator, if you know the brands in your database will be mutually exclusive and maintained by specified CSRs or administrators, you do not need to create an access group for each brand because access across brands is not required. You only set access permissions when you have one customer service representative (CSR) who administers multiple brands.

Use Access Privileges, a component of Business Configuration Center, to give BRM users access to a given brand or account groups. For more information on permissions and access groups, see "About Granting Access to Brands".

To enable access to top-level brands:

1. Log in to Configuration Center as the **root** user.

**Note:** If you are a brand administrator controlling access to a sub-brand in your brand, log in to Configuration Center by using your administrator login and password.

2. Click the icon for Access Privileges shown in Figure 19–4:

***Figure 19–4 Access Privileges Icon***

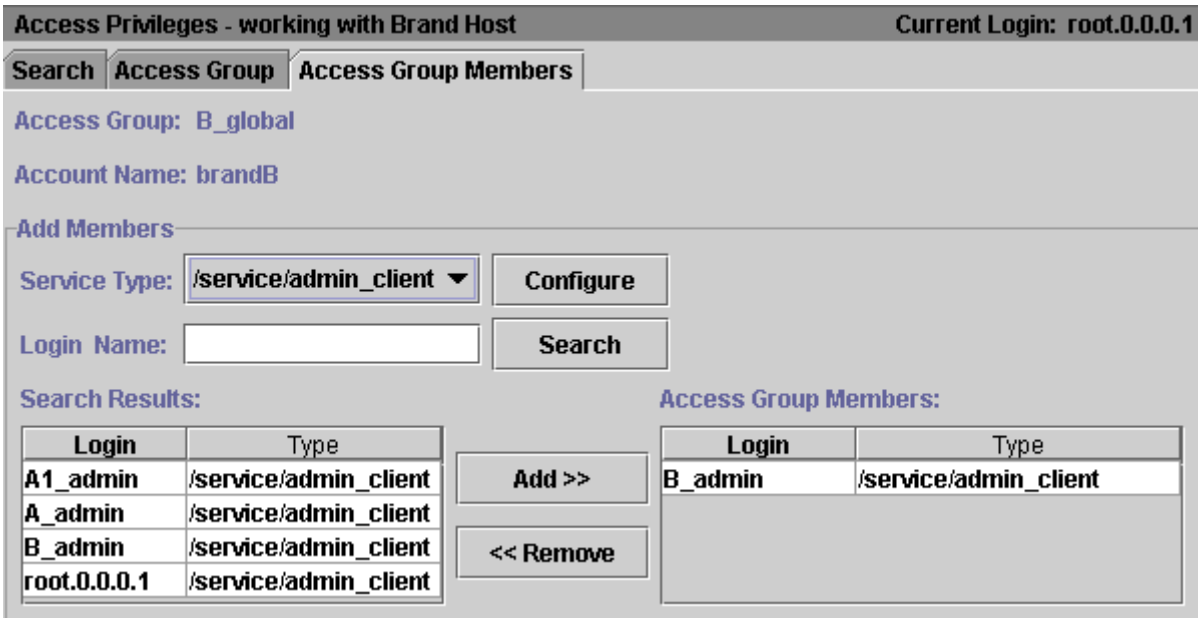
3. Click **Access Group** tab.
4. Enter a name and select one brand, sub-brand, or account group for the access group from the tree as shown in Figure 19–5. The following example creates the access group for the brand *brandB*:

***Figure 19–5 Access Group Tab***


5. Click **Save**.
6. Click the **Access Group Members** tab.
7. Search for login names belonging to the **admin_client** service.

   Figure 19–6 shows a search for all login names for the **admin_client** service.

***Figure 19–6 Access Group Members Tab***



**Tip:** You typically add logins for the **admin_client** service to an access group, so this service is available by default. If you want to add logins from another service, such as **pcm_client**, use the **Configure** button to add that service to the list of available services.

8. Select a login from the **Search Results** list to include in the access group, and then click **Add**.

   In this example, the login name **B_admin** now has access privileges to brandB when logging in under the **admin_client** service. This login name can create and modify customer accounts for Brand1 in Customer Center.

   You can add more login names now or later.

9. Click **Save**.

### Giving Users Access to All Accounts in BRM

Members of the root access group have access to all accounts in the database, regardless of whether the accounts exist in brands or not.

To add users to the root access group:

1. Log in to Configuration Center.
2. Click the icon for Access Privileges as shown in Figure 19–7:

***Figure 19–7 Access Privileges Icon***

3. Perform a wildcard search for an access group by using the asterisk (*).
4. Select the access group "ACL Group for Root Access."
5. Add users as required. See "Controlling Access to Top-Level Brands".

To give users access only to root-level accounts and not to brand accounts, see "Limiting Users to Accessing Root-Level Accounts Only".

For more information on the root access group, see "About the Root Access Group".

### Limiting Users to Accessing Root-Level Accounts Only

To give users access to root-level accounts and restrict them from accessing brand accounts, perform these steps:

1. Add the root accounts to an account group.
2. Create an access group for the account group.
3. Add the CSRs to the access group.

## Giving Permissions to the Brand Administrator

For brand administrators to enable CSRs to create accounts, charge credit cards, and perform other customer-related operations, the host administrator must first give them permission to set the CSR permissions.

To enable brand administrator permissions by using Customer Center:

1. Log in to Customer Center as **root** and open the brand administrator account.
2. Choose **Edit - Permission**.
3. Select a permission string from the table in the dialog box.
4. Select **/accounttool/permissions** and select **Read/Write**.
5. Give the brand administrator other necessary permissions and save the account; for example, you might give the brand administrator permission to credit an account.
6. Click **Change Entry**.
7. Give the brand administrator other necessary permissions and save the account; for example, you might give the brand administrator permission to credit an account.
8. Click **Save**.

For more information on giving permissions, see Customer Center.

## Setting Up a Customer Center Search Filter for Access Group Owners

You can enable a Customer Center search filter that restricts the account group owner search to only those accounts that are also access group owners, rather than querying and returning all account group owners in the entire brand. This improves performance of the BRM search operation.

To enable the search filter, set the **infranet.billinggroup.search** flag in the Customer Center **Infranet.properties** file to **owner**. When the search operation is performed, the PIN_FLD_FLAGS value in the PCM_OP_PERM_ACL_GET_SUBGROUPS opcode is set to **1**.

To search the entire hierarchy for access group owners, regardless of whether the account is also an account group owner, set the **infranet.billinggroup.search flag to 0.**

## Setting Up Invoicing to Support Brands

The BRM system contains a default invoice template that you can customize for each brand in a BRM system. Host administrators do not need to configure the default invoice functionality for branding; it is already brand aware. Brand administrators can access and modify invoice templates without any prerequisite configuration by the host administrator.

If a brand administrator does not customize the invoice template for a brand, BRM defaults to the parent brand template.

**Important:** If you make any changes to the invoice template or the events displayed in the template, you must stop and restart the CM.

See "Designing and Generating Invoices" in *BRM Designing and Generating Invoices* for more information on customizing templates.

**Note:** If the brands in your system require customized templates that differ significantly, you can modify the PCM_OP_INV_POL_FORMAT_INVOICE_HTML policy opcode to load brand-specific data into the brand templates. For example, by adding a brand's company address information to the _CUSTADDR_ tag, you can generate a custom invoice with each brand's corporate address.

### Determining the Events to Display in an Invoice

By default, invoices display all events with currency balance impacts greater than zero; however, the host administrator can determine which events to display by adding them to an event file and loading the file into the **/config/invoice_events** object. The BRM system contains a default event file named **event.file**, which contains routine billing events.

To enable brands to display different events in their invoices, you must modify the PCM_OP_INV_POL_PREP_INVOICE policy opcode.

## Setting Up the General Ledger to Support Brands

The **pin_glid** file is the configuration file that contains all your general ledger (G/L) ID definitions. It is loaded automatically when you install BRM.

To enable your brands to have their own private pricing structures and G/L accounts, you define a G/L segment name for each brand in the system in this file. Each brand in BRM is limited to the IDs defined in the **pin_glid** configuration file; however, brands are not required to use all of them or to use the same event mappings.

**Note:** G/L IDs are stored in a text file, and brand permissions cannot be set on them. Therefore, brand administrators should set operating system access permissions on the G/L ID file to deny other brands from reading and changing the data.

See "About Collecting General Ledger Data" in *BRM Collecting General Ledger Data* for more information on setting up G/L IDs.

## Setting Up Reports to Support Brands

To enable the brands in your system to run BRM Reports, you must install and configure Oracle Business Intelligence Publisher. See the following documents in *BRM Reports*:

- Reporting by Brand
- About BRM Reports
- Installing BRM Reports

## Setting Up Web Registration to Support Brands

To enable Web registration for multiple brands, you need to create a registration home page for each brand in your system. All other pages can be used for any brand.

After a customer logs in and creates an account, the BRM system knows which brand the customer belongs to when the customer logs in to the system again.

**Note:** Before you create Web pages for multiple brands, you need to define access privileges. See "Controlling Access to Top-Level Brands".

## Setting Up Payment Methods for Brands

Online payment processors require a *merchant ID* and *merchant number* (store name) to deposit your BRM-initiated payments in the correct account. Merchant numbers identify an account with a credit card processor. Your online payment processor assigns your merchant numbers, and you set the default number for your BRM system in the CM configuration file.

To define a unique merchant number for each brand, you must customize the **fm_cust_pol_prep_billinfo.c** source file and then add an entry in the credit card DM configuration file for each brand.

See "About BRM-Initiated Payment Processing" in *BRM Configuring and Collecting Payments* for more information.

**Note:** You must install, configure, and certify the BRM interface to a supported credit card service before you define the merchant numbers for brands. The interface to the credit card services is provided with the default BRM installation.

## Running Utilities with a Branded Database

Some BRM billing and configuration utilities can be used for managing separate brands. For example, you can run billing for a specific brand or run utilities to configure different default business policies for different brands.

**Note:** Not all utilities support branding; many do not. For more information, see the documentation for the utility you are using.

**Caution:** The host administrator should run billing for the entire database at the same time. Running billing for a particular brand is not recommended because it is very resource intensive and slows system performance.

To run a utility for a specific brand:

1. Open the configuration file for the utility. For example, to run billing for a specific brand, open the **pin.conf** file in *BRM_Home*/**apps/pin_billd**.
2. Replace the login name and password with the brand administrator's login name and password:

   ```
   - nap login_name brand_administrator_login
   - nap login_pw brand_administrator_password
   ```

3. Run the utility, or run the script that runs the utility.
4. When the utility has finished, repeat this procedure for each brand, or change the login name and password back to the default host login and password.

**Important:** You can create brand-specific configuration files; for example, **pin.conf.brand_a** and **pin.conf.brand_b**.

# Managing Permission by Using a Custom Application

You can limit which CSRs have access to branded customer accounts by using an Access Control List (ACL). For more information, see "About Granting Access to Brands" and "Setting Up Brands for Brand Administrators".

BRM stores information about each ACL in **/group/acl** objects in the BRM database. You manage or access data in **/group/acl** objects by using the Permissioning facilities module (FM) standard opcodes.

## Managing /group/acl Objects

You create, modify, and delete **/group/acl** objects by using the PCM_OP_PERM_ACL* opcodes. These opcodes validate data in the input flist and then call Group FM standard opcodes to create, add, modify, or delete data in a **/group/acl** object. For example, the PCM_OP_PERM_ACL_GROUP_ADD_MEMBER opcode adds members to a **/group/acl** object by calling the PCM_OP_GROUP_ADD_MEMBER opcode. For more information on Group FM standard opcodes, see "Managing Hierarchical Account Groups by Using Your Custom Application" in *BRM Managing Accounts Receivable*.

To manage **/group/acl** objects, you use the following Permissioning FM standard opcodes:

- To create a **/group/acl** object, use the PCM_OP_PERM_ACL_GROUP_CREATE opcode.
- To add a member to a **/group/acl** object, use PCM_OP_PERM_ACL_GROUP_ADD_MEMBER.
- To delete a member from a **/group/acl** object, use the PCM_OP_PERM_ACL_GROUP_DELETE_MEMBER opcode.
- To modify the attributes in a **/group/acl** object, such as the ACL name, brand or group account, or list of authorized service types, use the PCM_OP_PERM_ACL_GROUP_MODIFY opcode.

  **Important:** PCM_OP_PERM_ACL_GROUP_MODIFY overwrites the list of authorized services in the **/group/acl** object. When the PIN_FLD_PERMITTEDS array is included in the input flist, PCM_OP_PERM_ACL_GROUP_MODIFY deletes all services from the **/group/acl** object and adds the services from the input flist.

- To delete an entire **/group/acl** object, use the PCM_OP_PERM_ACL_GROUP_DELETE opcode.

  **Note:** Deleting an ACL does not delete the brand account or affect services. It only removes the ACL.

## Accessing /group/acl Data

Use the following opcodes to access data in **/group/acl** opcodes. Client applications typically use these opcodes to find all ACLs that a CSR belongs to or to determine which brand or account group to display.

- To find all ACLs to which a CSR or member belongs, use the PCM_OP_PERM_FIND opcode. See "Finding CSR Membership".
- To determine which brand or account group to display in a client application, use the PCM_OP_PERM_GET_CREDENTIALS opcode and the PCM_OP_PERM_SET_CREDENTIALS opcode. See "Displaying ACLs in Multi-Branded Applications".

### Finding CSR Membership

Use PCM_OP_PERM_FIND to find all ACLs to which a CSR belongs. This opcode returns a list of all groups and brands that a CSR is authorized to access, along with specific information about each ACL.

### Displaying ACLs in Multi-Branded Applications

When a CSR logs into a multi-branded application, the following occurs:

1. The application displays a list of available brands and bill units, along with the currently active brand or group.
2. The CSR selects one brand or bill unit to access.
3. The client application retrieves the connection scope for the selected brand or bill unit.
4. The client application displays only those accounts in the selected brand or bill unit.

For more information, see "Adding Branding to Your Application" in *BRM Developer's Guide*.

You use the following opcodes to determine which ACL to display in a client application:

- Use PCM_OP_PERM_GET_CREDENTIALS to retrieve all ACLs that are authorized for a particular client application, along with the currently active ACL.
- Use PCM_OP_PERM_SET_CREDENTIALS to set the connection scope for the specified ACL.

# 20

# Managing Brands

This chapter describes how to create and manage multiple brands in a single Oracle Communications Billing and Revenue Management (BRM) system. The tasks described in this section are generally done by a brand administrator after the host administrator configures the BRM database for brands.

You must have the Brand Manager component to create and use brands.

## Overview of Brand Administrator Tasks

The brand administrator handles all operations for the brand, including the following tasks:

- Creating a Price List for Brand Accounts
- Changing Account Defaults for a Brand
- Creating a Brand-Specific Web Page
- Defining a Brand-Specific Invoice
- Defining a Brand-Specific General Ledger System
- Defining Reports for a Brand
- Creating CSR Accounts for a Brand
- Creating Sub-Brands
- Accessing Branded Customer Accounts
- Processing Payments for Brand Accounts
- Inactivating or Closing a Brand
- Preventing Duplicate Brand Names
- Changing the Brand of an Account by Using a Custom Application

## Creating a Price List for Brand Accounts

You need to create a price list for your brand before you can create customer accounts or sub-brands.

To create a price list for brand accounts:

1. Create a price list in Pricing Center that includes a plan list of type **new** named **CSR**.

2. Use the brand administrator login name and password when you commit the price list to BRM. This saves the price list to the brand.

**Note:** You can see a brand administrator's login name by opening the brand in Brand Manager.

## Changing Account Defaults for a Brand

The customer service representative (CSR) accounts that you create in a brand inherit the default account settings of the parent brand. Before you create your CSR accounts, configure the brand with the default settings that you want the accounts to have to eliminate the need to change the settings at a later date.

You can change the following account settings:

- The payment method. The default is **prepaid**.
- The minimum and maximum amounts that a CSR can credit or debit an account.
- The minimum and maximum amounts that a CSR can adjust payments to an account.

For more information on changing account defaults, see "Setting Up Permissions in BRM Applications" in *BRM System Administrator's Guide*.

## Creating a Brand-Specific Web Page

You use Self-Care Manager to create Web pages for each brand you host, which enable customers to display and modify their account data. The host administrator must first install and configure Self-Care Manager for the BRM system. Then, brand administrators set up the Web pages for each brand.

For more information on creating a brand-specific Web page, see "Setting Up Customer Self Care with Self-Care Manager".

## Defining a Brand-Specific Invoice

You can design a different invoice template for each brand. Invoice templates are HTML files or XSL style sheets. You then use **pin_load_invoice_template** to load the HTML or XSL invoice template for each brand. If a brand does not have a template associated with it, BRM uses the default system template.

**Note:**

- A brand can contain only one template.
- You can create a separate template for each brand, but not for an access group.

See "Designing and Generating Invoices" in *BRM Designing and Generating Invoices*.

## Defining a Brand-Specific General Ledger System

Accounts that exist in a brand are automatically assigned a general ledger (G/L) segment for that brand.

In a branded environment, the only G/L IDs you can use are those defined by the host administrator when the BRM database is configured. You are limited to these IDs, but you can map them to any G/L accounts in your external system.

For more information on defining a brand-specific general ledger system, see "About Collecting General Ledger Data" in *BRM Collecting General Ledger Data*.

## Defining Reports for a Brand

Using the BRM reports functionality, brand managers can generate brand-specific customer activity and billing reports such as accounts receivable reports, credit card reports, customer usage reports, and general ledger reports.

> **Note:** Before you can define reports for your brand, the host administrator must install and configure Oracle Business Intelligence Publisher. See "Installing BRM Reports" in *BRM Reports*.

The BRM Reports package includes brand-configured versions of the BRM base reports.

## Creating CSR Accounts for a Brand

You use Customer Center to create CSR accounts for brands and to define change permissions for CSRs.

For more information, see Customer Center Help.

To create CSR accounts for a brand:

1. Log in to Customer Center by using the brand login name and password.
2. Create the CSR account.

> **Note:** Select the CSR plan to add the **admin_client** service to the account.

3. Give the CSR permissions. See "Setting Up Permissions in BRM Applications" in *BRM System Administrator's Guide*.
4. Add the CSR to the access group for each brand you want the CSR to access.

> **Note:** Before brand administrators can give account permissions to CSRs, the host administrator must give brand administrators permission to set account permissions. See "Giving Permissions to the Brand Administrator".

For more information on CSRs, see "About Managing Customers" in *BRM Concepts*

The CSR account appears in Customer Center as a child account of the brand administrator account. If you want, you can move the CSR account to the same level as the administrator account by dragging the account icon.

# Creating Sub-Brands

To create a sub-brand, log in to Configuration Center by using the brand administrator login name and password.

See "Creating Brands" for detailed instructions on creating sub-brands.

**Note:** There is no limit to the number of sub-brands you can create; however, sub-brands in the same brand hierarchy must have unique names. Sub-brands in different brand hierarchies can share the same name.

For example, the brand East Coast Enterprises cannot have two sub-brands named Sales Division. However, it can have a sub-brand named Joanne's Toys, and each of these brands can contain a sub-brand named Sales Division.

## Controlling Access to Sub-Brands and Account Groups

You control access to a sub-brand and account groups in a brand by creating an access group. You create access groups by using Access Privileges, an application in Configuration Center.

**Note:** You must have the Branding component of BRM to use Access Privileges.

See "Controlling Access to Top-Level Brands" for instructions.

# Accessing Branded Customer Accounts

Using Customer Center, CSRs and brand administrators can create and modify accounts for any brand or account group for which they have access privileges.

**Note:**

- When a CSR logs in to either tool, the word *Self* appears in parentheses after the name of the brand to which the CSR's BRM account belongs. A CSR might have access privileges to multiple brands, but the CSR account belongs to only one brand.
- By default, CSRs can modify only the customer accounts they create; they cannot access accounts created by other CSRs. To enable CSRs to access all accounts in a brand, you must add them to the access group for the brand.
- Brands appear in Customer Center as regular accounts, but they are identified by the account number and name of the billing contact, not by the brand name.

## Accessing Branded Customer Accounts in Customer Center

CSRs that have access to multiple brands can access accounts in all brands at the same time, without having to close one brand and open another.

To access a branded customer account:

1. Log in to Customer Center by using the brand login name and password.
2. Click **Search,** which opens the Search Accounts dialog box.
3. From **Brand Selector**, choose the brand to open.
4. Enter the search criteria.
5. Click **Search**.

> **Note:** You cannot move a customer account to a different brand. You must first close the account in the old brand and then recreate the account in the new brand.

## Processing Payments for Brand Accounts

You use Payment Tool to process payments, refunds, and payment reversals for brand accounts.

To process payments for brand accounts:

1. Log in to Payment Tool as a brand administrator or as a CSR with access privileges.
2. Click **Select Brand** and choose the brand to work in.
3. Process payments as you normally would.

See Payment Tool Help for details.

## Inactivating or Closing a Brand

To inactivate or close a brand, change the status of the brand account in Customer Center or in Brand Manager. You cannot delete a brand from the BRM system.

Inactivating or closing a brand changes the status of the brand and all customer accounts that belong to that brand; it does not change the status of any sub-brands.

> **Note:** When you reactivate a brand, all customer accounts in the brand are also reactivated, regardless of whether they were inactivated or closed independently from the brand.

See "About Closing, Inactivating, and Reactivating Brands" for more information.

## Preventing Duplicate Brand Names

To prevent users from creating duplicate brand names within a brand and its sub-brands (the default) or within a BRM system, set the **check_unique** flag in the PCM_OP_CUST_POL_SET_BRANDINFO policy opcode. To allow duplicate brand names, disable this flag.

## Changing the Brand of an Account by Using a Custom Application

To change the brand of an account, use PCM_OP_CUST_SET_BRANDINFO. This opcode updates the brand **/account** object.

The PCM_OP_CUST_COMMIT_CUSTOMER opcode calls the PCM_OP_CUST_SET_BRANDINFO opcode when a brand account is created.

PCM_OP_CUST_SET_BRANDINFO calls the PCM_OP_CUST_POL_SET_BRANDINFO policy opcode for any user customizations, such as preventing duplicate brands.

To customize brand assignment, use the PCM_OP_CUST_POL_SET_BRANDINFO policy opcode. This opcode allows you to customize the error information or information about the brand returned to PCM_OP_CUST_SET_BRANDINFO. the PCM_OP_CUST_POL_SET_BRANDINFO policy opcode also includes hooks for setting site-specific information.

# Part III

## Migrating Customer Accounts

Part III describes how migrate customer accounts from legacy databases. It contains the following chapters:

# 21

# Understanding Conversion Manager

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) Conversion Manager. It describes the process of converting data from a legacy database to the BRM database.

**Important:** Conversion Manager is an optional component, not part of base BRM, and requires a separate license.

## About Conversion Manager

You use Conversion Manager to load legacy data into BRM. Conversion Manager supports the following types of data:

- Account data, such as the customer's name, address, profile, and discount information.
- Service data, such as the services subscribed to by the accounts.
- Product data, such as products purchased by the accounts.
- Billing data.
- Account hierarchy data.
- Balance data, including data for all resources. When data is migrated, rollover events can use the balance data to manage rollovers.

You can load account and balance data together, or you can load account data first and balance data second. All account data for an account must be loaded before loading balance data.

Before migrating data, do the following:

- Create a price list that applies to the migrated data before migrating account and balance data. Migrated data needs to reference the price list objects in the BRM database.
- Make sure that cycle arrears fees, billing-time discounts, and rollovers have been processed in the legacy system.

When the accounts are migrated, cycle forward fees and discount grants (for example, free minutes) are applied. However, all delayed events are handled in the current bill.

**Note:** The following features are not supported by Conversion Manager:

- Brand management
- Auditing

## Overview of the Data Conversion Process

Converting legacy data includes these tasks:

- Understanding the data in the legacy system and deciding how to convert it to the BRM database.
- Mapping the data in the legacy database to the BRM database. To do so, you create XML files, which are validated by the Conversion Manager XSD schema files.
- Migrating the data to the BRM database by using the **pin_cmt** utility. To migrate data:
  - Import data into the BRM database. The data is hidden from BRM processes.
  - Deploy the staged data to the production area.

  See "Loading Legacy Data into the BRM Database" for more information.

**Important:** BRM must be running when you import and deploy data.

## About Testing Your Data Mapping

You test your data mapping by using a test database. To test the data mapping:

1. Create a subset of the data in an XML file. See "Mapping Legacy Data to the BRM Database".
2. Load the data into the test database. See "Loading Legacy Data into the BRM Database".
3. Test the data. See "Testing the Imported Data".

After you determine that the data mapping works, you import the data into the production database.

## About Mapping Data

You convert legacy data to XML format by using an extraction utility.

**Important:** Conversion Manager does not include a utility for extracting legacy data into XML files. You need to develop your own extraction utility or obtain it from third-party sources.

Conversion Manager converts data from any type of legacy database; for example, Oracle.

To map data from the legacy database to the BRM data schema, you need a thorough understanding of the legacy data and how BRM stores data.

If your legacy data includes data not currently supported in BRM, you can create new storable classes, or extend storable classes, to support the custom data. You can then extend the Conversion Manager XML schema to migrate the data. See "Migrating Data by Using New and Extended Storable Classes".

You can configure data enrichment to allow system-wide values to be bulk inserted and to provide batch controls for operational integrity.

## About Loading Data

You use the **pin_cmt** utility to load the converted data into the BRM database. Loading data includes two processes: importing accounts and deploying accounts.

- First, you *import* the data into the BRM database. When the data is imported, it is hidden from BRM processes, so it is not part of your production system. (Data is hidden by changing the database number section in the Portal object ID (POID) of each imported object. The database number is set to a number defined in the **infranet.cmt.dbnumber** entry in the **Infranet.properties** file for the **pin_cmt** utility; for example, **0.0.0.12**.)

  When you import data, you specify a *stage ID*. This enables you to load data in stages; for example, you can load a specific type of account first. A single database schema can include multiple stages.

- After the data is imported, you *deploy* the data. When you deploy data, the imported data is exposed as production data.

  You control which accounts are deployed by entering the stage ID that you used when importing the data. In addition, you specify the billing day of month (DOM). Therefore, only those accounts with the specified stage ID and DOM are deployed.

  After the data is deployed, the database number in the POID is changed to the actual database number, and the data becomes available for use by BRM processes.

  **Note:**

  - Events that apply to a migrated account are *not* processed until the account is deployed.
  - When data is deployed, the database number used is defined in the **infranet.cmt.targetdbnumber** entry in the **pin_cmt** utility **Infranet.properties** file. By default, the number is **0.0.0.1**.
  - File processing data, such as the file name and batch ID, is stored in the **/batch/cmt** object.

  When accounts are deployed:

  – The bill cycles are started.

  – Cycle fees are applied.

  – If you use a multischema system, the uniqueness table in the primary database schema is updated.

  **Important:** After deploying your imported accounts, stop and restart Pipeline Manager to send new account data to Pipeline Manager.

### About Verifying Data before It Is Deployed

You can look at data that is imported but not yet deployed by logging in to Customer Center connected to a scoped Oracle Data Manager (DM). See "Viewing Data before Deploying".

### About Migrating Data to Multischema Systems

To load data into a multischema system, run a separate instance of the **pin_cmt** utility for each database schema. The **pin_cmt** utility connects to the primary schema and creates a **/uniqueness** object for every account and service migrated to BRM.

**Important:** If you have a multischema system, make sure the stage IDs are larger than the database IDs for the schemas. For example, if you have a schema with the number 0.0.0.5, use stage IDs larger than 5.

## About Loading Data by Using Multiple Files

You can use multiple XML files to load data. For example, you can load account information separately from balance data. You can use the same staging area for multiple files.

**Important:**

- Load account data before loading balance data.
- Load parent accounts before loading child accounts.

## About Reloading Data

If the **pin_cmt** utility runs out of space in your BRM database for data rows, the loading process stops. Data that is not loaded can be loaded after more space is made available in the database. See "Reloading Data".

# 22

# Installing and Configuring Conversion Manager

This chapter describes how to install Oracle Communications Billing and Revenue Management (BRM) Conversion Manager.

Before you read this document, you should be familiar with how Conversion Manager works. See "Understanding Conversion Manager".

> **Note:** Conversion Manager is an optional feature that requires a separate license.

## System Requirements

Conversion Manager is available for the Solaris, HP-UX IA64, AIX, and Linux operating systems and for the Oracle database.

### Software Requirements

Before installing Conversion Manager, you must install:

- Third-Party software, which includes the libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- Oracle *10g*, Oracle *11g*, or Oracle *12c*.
- **libocijdbc11.so**. Conversion Manager requires the use of this OCI Instant JDBC Library.

### Information Requirements

You need the following information about your existing BRM system during the Conversion Manager installation:

- Database alias name of the staged database.
- Database port and number for the staged database.
- Database user name and password for the staged database.
- Database alias name of the primary database schema.
- Database port and number for the primary database schema.
- Database user name and password for the primary database schema.

# Installing Conversion Manager

> **Note:** If you have already installed the product, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

To install Conversion Manager:

1. Download the software to a temporary directory (*temp_dir*).

   > **Important:**
   >
   > - If you download to a Windows workstation, use **FTP** to copy the **.bin** file to a temporary directory on your UNIX server.
   > - You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid "Out of Memory" error messages in the log file. For more information, see "Increasing Heap Size to Avoid "Out of Memory" Error Messages" in *BRM Installation Guide*.

2. Go to the directory where you installed the Third-Party package and source the **source.me** file.

   > **Note:** You must source the **source.me** file to proceed with installation, otherwise "suitable JVM not found" and other error messages appear.

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

3. Go to temp_dir and enter this command:

   ```
   7.5.0_ConversionMgr_platform_opt.bin
   ```

   where *platform* is the operating system name.

   > **Note:** You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

4. Follow the instructions displayed during installation. The default installation directory for Conversion Manager is **/opt/portal/7.5**.

> **Note:** The installation program does not prompt you for the installation directory if BRM or Conversion Manager is already installed on the computer and automatically installs the package at the *BRM_Home* location. *BRM_Home* is the directory in which you installed the BRM software.

5. Go to the directory where you installed the Conversion Manager package and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

6. Go to the *BRM_Home*/**setup** directory and run the **pin_setup** script.

> **Note:** The **pin_setup** script starts all required BRM processes.

Your Conversion Manager installation is now complete.

# Configuring the pin_cmt Utility

You use the **pin_cmt** utility to migrate the data. To configure performance, log file, and other settings for this utility, edit the **Infranet.properties** file in *BRM_Home*/**apps/cmt**.

> **Note:** Many of the entries in the **Infranet.properties** file include default settings that are applicable for testing your data mapping; for example, the number of threads that the **pin_cmt** utility uses. You can change the settings when you load data into the production system.

In addition, you need to edit the **pin.conf** file for the **cmt_mta_cycle_fees** utility. This utility is run automatically by the **pin_cmt** utility to apply cycle fees. The **pin.conf** file is in *BRM_Home*/**apps/cmt**. It includes the standard connection parameters and multithreaded application (MTA) performance setting. You can also specify to not apply cycle fees to accounts.

See "Applying Cycle Fees to Deployed Accounts".

## Referencing JAR Files

To ensure that the **pin_cmt** utility finds all the necessary JAR files, include them in the **pin_cmt** file, in the line that invokes Java. The format is:

```
BRM_Home/jre/bin/java -Dfile.encoding=utf-8 -cp "jar_A:jar_B:jar_C:jar_D" com.portal.cmt.Cmt $1 $2
$3 $4 $5 $6 $7 $8 $9
```

where *jar_A*:*jar_B*:*jar_C*:*jar_D* is the list of Java class JAR files. Oracle library references are appended at the end.

Include all entries on one line with no line breaks. Separate JAR entries with a colon. Do not include any spaces in the list of JAR files. The number of elements is limited only by command-line length (approximately 8 kilobytes on most machines).

The following shows an example:

```
BRM_Home/jre/bin/java -Dfile.encoding=utf-8 -cp "BRM_Home/apps/cmt/cmt.jar:BRM_
Home/jars/xercesImpl.jar:BRM_Home/jars/xmlParserAPIs.jar:BRM_Home/jars/pcm.jar:BRM_
Home/jars/pcmext.jar:BRM_Home/apps/cmt" com.portal.cmt.Cmt $1 $2 $3 $4 $5 $6 $7 $8 $9
```

## Defining Timestamp Validation for Finite Partitioned Classes

When you import data that is in finite partitioned classes, you can define the timestamp validation mode that is encoded in the POID of a finite partitioned class. You provide the timestamp validation mode as the value for the **infranet.cmt.timestampvalidation** entry in the **Infranet.properties** file for use by Conversion Manager.

Table 22–1 lists the valid entries for **infranet.cmt.timestampvalidation**.

***Table 22–1 Valid Modes for Timestamp Validation***

| Input Value | Timestamp Validation Mode |
|---|---|
| 0 | No timestamp validations are performed. If an entry is provided in **created_t**, that creation time is used for a finite partitioned class. Otherwise, the current system time is used for a finite partitioned class. |
| 1 | The **created_t** entry is mandatory for any finite partitioned class. It should be provided in the input XML file containing the legacy data to be imported. If **created_t** is not provided, Conversion Manager generates an error.<br>This is the default value for **Infranet.cmt.timestampvalidation**. |
| 2 | The default time is used for the POID encoding. If **created_t** is provided for any finite partitioned class, Conversion Manager generates an error. |

To add the **infranet.cmt.timestampvalidation** entry to the **Infranet.properties** file:

1. Open the *BRM_Home*/**apps**/**cmt**/**Infranet.properties** file in a text editor.
2. Add the following entry. Provide the required value for *n*. See Table 22–1.

   ```
   infranet.cmt.timestampvalidation = n
   ```

   **Note:** If you provide an invalid input for **infranet.cmt.timestampvalidation**, BRM replaces that value with the default value (**1**).

3. Save and close the **Infranet.properties** file.

## Viewing Data before Deploying

When you import data, the data is migrated to your production database, but it is not accessible to normal BRM processes until it is deployed. To view the data, you need to configure one or more Oracle Data Manager (DMs) as scoped DMs. A *scoped DM* can view accounts according to their stage ID. You can configure multiple scoped DMs, one for each staging area.

In addition to configuring scoped DMs, you also need to configure the production Oracle DM to support DM scoping.

A BRM system that includes scoped DMs is called a *scoped system*.

To configure a scoped system:

1. Run the **pin_convert_nonunique.sql** script in *BRM_Home***/apps/cmt/scripts**. This script drops unique constraints for indexes used in account creation.

   **Note:** By default, the **pin_convert_nonunique.pl** script uses PINX00 to identify tablespaces for indexes. If you use a name other than PINX00, edit the **pin_convert_nonunique.pl** script to change PINX00 to your tablespace name.

2. Run the *BRM_Home***/sys/dd/data/cmt_copy_group_planlists_oracle.sql** stored procedure. This procedure duplicates the **/group/plan_list** object with the staged database number, which enables you to create accounts by using Customer Center.

   The stored procedure uses the following parameters:

   - Database number; for example, 5
   - Return code (number)
   - Return message (varchar2(100))

   The following shows a sample call using PL/SQL:

   ```
   declare
   p_return_code number;
   p_return_mesg varchar2(100);
   begin
   CMT_COPY_GROUP_PLANLISTS_ORACLE(5,p_return_code,p_return_mesg);
   end;
   ```

3. Create a **pin.conf** file for each scoped Oracle DM. To do so, edit the Oracle DM **pin.conf** file in *BRM_Home***/sys/dm_oracle**.

   Edit the following entries:

   - Set the **scoped_system** entry to **1**. This enables database scoping.
   - Set the **production_system** entry to **0**. This identifies the DM as a scoped DM.
   - Set the **staging_db_number** entry to the stage ID used for importing data. If you use a multischema system, the stage ID must be larger than the largest schema database number.
   - Include the table names for the data you are importing in the **scoped_exception_tables** entry.

   For example, a scoped DM for stage ID 11 has the following entries:

   ```
   - dm scoped_system 1
   - dm production_system 0
   - dm staging_db_number 11
   - dm scoped_exception_tables channel_event_t channel_t config_t data_t deal_t
   plan_t product_t rate_plan_selector_t rate_plan_t rate_t search_t zonemap_t
   ```

4. Repeat the DM **pin.conf** configuration for each scoped DM, using a different stage ID for each one.

5. Configure the production Oracle DM by editing the following entries:

   - Set the **scoped_system** entry to **1**.

- Set the **production_system** entry to **1**. This identifies the DM as the production DM.
- Include the table names for the data you are importing in the **scoped_exception_tables** entry.

The configuration for the production system looks like this:

```
- dm scoped_system 1
- dm production_system 1
- dm scoped_exception_tables channel_event_t channel_t config_t data_t deal_t
plan_t product_t rate_plan_selector_t rate_plan_t rate_t search_t zonemap_t
```

6. Start the scoped DMs.
7. Create a root account for each stage ID:
   a. Start the scoped Oracle DMs.
   b. Open the **load_pin_acct** script in *BRM_Home*/**apps/cmt/scripts**.
   c. Edit the first line of the script to point to a valid Perl path. For example:

      ```
      /opt/portal/ThirdParty/perl/5.8.0/bin
      ```

   d. Edit the **$dm_port** entry to point to the Oracle DM port number. For example:

      ```
      $dm_port = 22251
      ```

   e. Make sure the **$db_no** entry points to the production database. For example, if the production database number is 0.0.0.1, and the staged database number is 0.0.0.11, use the following entry:

      ```
      $db_no = 0.0.0.1
      ```

   f. Save and close the file.
   g. Run the **load_pin_acct** script using the following command:

      ```
      load_pin_acct -I pin_init_acct
      ```

      **Note:** The input flist used when the root account is created still refers to the production database number because the staging DM translates it to the staging database number.

8. Stop and restart all Connection Managers and Oracle DMs used for accessing the staged data.

## Enabling Multischema Loading

If you are migrating data to a multischema system, edit the entries shown in Table 22–2 in the *BRM_Home*/**apps/cmt/Infranet.properties** file:

***Table 22–2 Infranet.properties File Entries to Enable Multischema Loading***

| Entry | Description |
|---|---|
| **infranet.cmt.deploy.multidb** | Enables or disables data migration in a multischema system. Enter **true** or **false**. |
| **infranet.cmt.primarydbname** | Specifies the database schema name (for example, **pindb1**). |

*Table 22–2 (Cont.) Infranet.properties File Entries to Enable Multischema Loading*

| Entry | Description |
|---|---|
| **infranet.cmt.primarydbuserid** | Specifies the schema login name. |
| **infranet.cmt.primarydbpasswd** | Specifies the schema login password. |
| **infranet.cmt.primarydbnumber** | Specifies the database number for the primary schema (for example, **0.0.0.2**). |

## Enabling XML Validation

The **pin_cmt** utility can validate the input files to ensure that the data has the correct structure. However, this decreases performance. If you validate the XML prior to loading the data, you can leave validation disabled.

> **Tip:** You can test the data mapping by enabling validation when you migrate data to a test database. You can also test the data mapping by using the Conversion Manager XSD file with an online XML validator or third-party tool.

To enable or disable XML validation, edit the **infranet.cmt.preprocess.validation** entry in the **Infranet.properties** file:

```
infranet.cmt.preprocess.validation = true
```

## Configuring the Database Setup

Edit the following entries in the **Infranet.properties** file to match your typical account configuration:

- Use the **infranet.cmt.avgnoofservices** entry to specify the average number of services that each account owns. This entry reserves Portal object IDs (POIDs) for the objects that will be created by the **pin_cmt** utility. The default is **5**.
- Use the **infranet.cmt.avgnoofdevices** entry to specify the average number of devices that each account owns. This entry reserves POIDs for the objects that will be created by the **pin_cmt** utility. The default is **2**.
- Use the **infranet.cmt.noofrecords** entry to specify the average number of account records in each input XML data file.
- Use the **infranet.cmt.dbnumber** entry to set the staged database number (for example, **0.0.0.12**). Use the **infranet.cmt.dbname**, **infranet.cmt.userid**, and **infranet.cmt.passwd** entries for the rest of the staged database information.
- Use the **infranet.cmt.targetdbnumber** entry to set the target database.

> **Note:** If an account owns more or fewer services or devices than the numbers you specify, the objects are still loaded.

## Setting the Default Credit Limit Profile

When accounts are imported, balance groups are created for the accounts.

You can specify which credit limit profile is assigned to each balance group in the following ways:

- You can specify the default credit limit profile by editing the **infranet.cmt.creditprofile** entry in the **Infranet.properties** file:

  ```
  infranet.cmt.creditprofile = IndexID
  ```

  Each credit limit profile is stored in a PIN_FLD_PROFILES array in the **/config/credit_profile** object. Replace *IndexID* with the index ID of the appropriate PIN_FLD_PROFILES array. In this example, you would replace *IndexID* with **3**:

  ```
  0 PIN_FLD_PROFILES                ARRAY [3] allocated 3, used 3
  1    PIN_FLD_CREDIT_FLOOR         DECIMAL [0] NULL
  1    PIN_FLD_CREDIT_LIMIT         DECIMAL [0] 133
  1    PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
  ```

- You can assign a credit limit, credit floor, and credit threshold to each balance group when you run the **pin_cmt** utility. To do so, add the following XML elements to the input XML file that you import with **pin_cmt**:
  - **<CrdLmt>**
  - **<CrdFlr>**
  - **<CrdTrsh>**

  The values specified in the input XML file take higher precedence than the value of the **infranet.cmt.creditprofile** entry in the **Infranet.properties** file.

## Applying Cycle Fees to Deployed Accounts

By default, the **pin_cmt** utility runs a separate utility, **cmt_mta_cycle_fees**, to apply cycle fees after deploying accounts. To disable this, edit the **infranet.cmt.deploy.opcode** entry in the *BRM_Home***/apps/cmt/Infranet.properties** file:

```
infranet.cmt.deploy.opcode = false
```

## Migrating New Balances to an Account Without Deleting Existing Balances

By default, when you migrate new balances to an account that was previously migrated using Conversion Manager, the **pin_cmt** utility deletes the existing balances on the account and associates only the newly migrated balances to the account.

To migrate new balances to an account without deleting the existing balances:

1. Open the *BRM_Home***/apps/cmt/Infranet.properties** file in a text editor.
2. Add the following entry:

   ```
   infranet.cmt.deleteexistingbalances = false
   ```

3. Save and close the file.

## Supporting 31-Day Billing

If your BRM implementation uses 31-day billing, configure the following entries in the **Infranet.properties** file:

```
infranet.cmt.31daybilling = true
infranet.cmt.31daybilling.forward = true
```

If the **infranet.cmt.31daybilling.forward** entry is enabled, BRM performs forward billing if the billing day of month is greater than 28. If it is disabled, BRM performs backward billing.

For more information, see "Using 31-Day Billing" in *BRM Configuring and Running Billing*.

### Supporting Delayed Billing

If your BRM implementation uses delayed billing, configure the following entry in the **Infranet.properties** file:

```
infranet.cmt.billingdelay=true
```

For more information, see "Setting Up Delayed Billing" in *BRM Configuring and Running Billing*.

## Improving Conversion Manager Performance

Conversion Manager performance depends on the data and features used for a given BRM implementation. Therefore, this documentation includes guidelines, but does not provide specific values for performance-related configuration settings.

### XML File Formatting

Limit the number of accounts in an XML file to 10,000 accounts. Conversion Manager supports XML files with any number of accounts, but limiting the number of accounts enables you to debug the migration in the case of an error message.

When you create the XML files:

- Validate the files with the Conversion Manager physical XML schema.
- Use the proper tab (\t) and new line (\n).
- Do not use spaces.
- Make sure all the required tags are present in the XML.

### Running Multiple Instances of the pin_cmt Utility

Use multiple instances of the **pin_cmt** utility to load multiple XML files simultaneously.

For example, you can write a batch/shell script similar to this:

```
cat >> run.sh << EOF
date > start.txt
pin_cmt -import -file one.xml 1 &
pin_cmt -import -file two.xml 1 &
pin_cmt -import -file three.xml 1 &
pin_cmt -import -file four.xml 1 &
pin_cmt -import -file five.xml 1 &
wait
date > end.txt
EOF
```

While running the script, measure the CPU load and time taken by the **pin_cmt** utility for this entire batch. You can then increase the number of **pin_cmt** instances until the CPU load reaches 90%.

## Using Connection Pooling with Conversion Manager

To improve performance, you can use connection pooling with Conversion Manager. Configure the following entries in the **Infranet.properties** file:

- **infranet.cmt.minconns**
- **infranet.cmt.maxconns**
- **infranet.cmt.timeout**

For example:

```
infranet.cmt.minconns = 15
infranet.cmt.maxconns = 30
infranet.cmt.timeout = 1000
```

For more information, see "About Connection Pooling" in *BRM System Administrator's Guide*.

## Configuring Log File Levels

For test migrations, the default log file settings are set to return troubleshooting information. For better performance, you should change those settings to report less information.

- In the *BRM_Home*/**apps/cmt/pin.conf** file, change the **pin_mta loglevel** entry to **1**.
- In the *BRM_Home*/**apps/cmt/Infranet.properties** file, change the **infranet.log.level** entry to **1**.

## System Resources

Migrating data can use a lot of system resources. Performance can depend on:

- RAM size
- Number of CPUs
- Disk quotas
- File system usage

## Conversion Manager Preload Tuning

After you have validated the XML files and are ready to load data, edit the *BRM_Home*/**apps/cmt/Infranet.properties** file to maximize performance.

- Turn off XML validation:

  ```
  infranet.cmt.preprocess.validation = false
  ```

- Set the following entries to correspond to the amount of data in the XML files. The **infranet.cmt.noofrecords** entry specifies the number of accounts.

  ```
  infranet.cmt.noofrecords      = 1000
  infranet.cmt.avgnoofservices  = 5
  infranet.cmt.avgnoofdevices   = 2
  ```

- Configure sufficient threads in the preprocess thread pool.

  ```
  infranet.cmt.preprocess.num_of_threads = 5
  ```

  The default value is **5**, which is the optimal value in most cases.

- Make sure there are enough connections in the staging database connection pool:

```
infranet.cmt.minconns   = 15
infranet.cmt.maxconns   = 30
infranet.cmt.timeout    = 1000
```

### Increasing Memory Allocation to Prevent a System Hang

To test your mapping, you can run the **pin_cmt** utility with the default memory allocation. However, when migrating a production database, you might need to increase the memory allocation to prevent a system hang.

Edit the **pin_cmt** script to avoid system hangs.

> **Note:** Depending on your system configuration and load, your settings might be different.

```
JAVA_OPTIONS="-Xms1024m -Xmx2048m"
BRM_Home/jre/bin/java $JAVA_OPTIONS -Dfile.encoding=utf-8 -cp "BRM_Home/apps/cmt/cmt.jar:BRM_
Home/jars/xercesImpl.jar:BRM_Home/jars/xmlParserAPIs.jar:BRM_Home/jars/pcm.jar:BRM_
Home/jars/pcmext.jar:BRM_Home/apps/cmt"com.portal.cmt.Cmt $1 $2 $3 $4 $5 $6 $7 $8 $9
```

## Conversion Manager Load Tuning

In the *BRM_Home*/**apps/cmt/Infranet.properties** file, verify the command line parameters for the SQL*Loader.

```
infranet.cmt.sqlldr = sqlldr streamsize=2621440 readsize=2621440 columnarrayrows=5000
```

When you use the flag **direct="true"**, make sure the BRM user has **DBMS_LOCK** permission.

## Conversion Manager Deploy Tuning

Before you deploy data, edit the following entries in the Conversion Manager pin.conf file. The file is located in *BRM_Home*/**apps/cmt.**

- Set the log level to 1:

```
- pin_mta loglevel 1
```

- Specify the appropriate capacity settings for your system; for example:

```
- pin_mta   children     5
- pin_mta   per_batch    500
- pin_mta   per_step     1000
- pin_mta   fetch_size   5000
```

# 23

# Mapping Legacy Data to the BRM Database

This chapter explains, in general terms, what you need to understand about your legacy data to successfully migrate it to an Oracle Communications Billing and Revenue Management (BRM) database.

**Note:** Describing your legacy data in detail is beyond the scope of this chapter because each system is unique.

## About Creating XML Files

To migrate data, you create XML files that contain the data from your legacy database. You then migrate the data to the BRM database by using the **pin_cmt** utility.

Sample XML files for each type of data are available in *BRM_home***/apps/cmt/sample_data**. *BRM_home* is the directory in which BRM is installed.

## About the XSD Files

Conversion Manager includes two sets of XSD files:

- A set of *conceptual* files that you can use for mapping legacy data to BRM objects. These files are easier to read than the set of files used for converting data.
- A set of *physical* files that are used by Conversion Manager when converting data. These files are not as easy to read, but they are written in a way that optimizes performance.

The difference between the types of files is how the XML tags are named. In the conceptual files, the XML tags use the same field names as the BRM object fields.

For example:

- In the conceptual files:

  PIN_FLD_ACCOUNT_NO = <FldAccountNo>

- In the physical files:

  PIN_FLD_ACCOUNT_NO = <ActNo>

To set up mapping, you can use the easily-readable conceptual files to determine how to map data. Then you can edit the physical files, after you know how the data is mapped.

The conceptual files are in *BRM_home***/apps/cmt/schema_files/conceptual**.

The physical files are in *BRM_home***/apps/cmt/schema_files/physical**.

Use the following physical XSD files to create your XML files:

- Use the **CMT_Subscriber.xsd** file for subscriber data.
- Use the **CMT_Balances.xsd** file for balance data.

**Tips:**

- Before you migrate data, validate the XML files with the physical XSD files.
- Do not use spaces while generating the XML files. Use tabs (**\t**) and new lines (**\n**).

# About the Types of Data to Convert

Table 23–1 shows the types of legacy data that can be loaded into the BRM database:

***Table 23–1 Legacy Data Storable in BRM Database***

| Type of Data | Description |
|---|---|
| **Account data** | Subscriber data such as name, address, profile, current account balances, group charge sharing, and account hierarchy. |
| **Product, discounts, and deals** | Deals purchased by the customer and associated with the account. |
| **Bill** | Bill information such as billing cycle and balances from the legacy billing system. |
| **Services** | Data pertaining to services owned by the account, such as a wireless data service.<br>For example, when converting data for a wireless data service, the following data is also converted:<br>■ Device SIM<br>■ Device number |

# Tables Affected by the Conversion Process

Table 23–2 shows the BRM tables that are affected by the conversion process. For more information on the BRM tables and the fields in each table, see the following:

- "Storable Class-to-SQL Mapping" in *BRM Developer's Reference*
- "Storable Class Definitions" in *BRM Developer's Reference*

**Note:** The database tables are created during BRM installation.

***Table 23–2 Tables Affected by the Conversion Process***

| Table Name | Description |
|---|---|
| **ACCOUNT_T** | Master account table that represents billable accounts in BRM. Only one row is added to this table for each account in BRM. |
| **ACCOUNT_EXEMPTIONS_T** | Contains an entry for each tax exemption that applies to an account. |

*Table 23–2 (Cont.) Tables Affected by the Conversion Process*

| Table Name | Description |
|---|---|
| **ACCOUNT_NAMEINFO_T** | Contains name and address information for accounts. The table contains one row for each name and address type (home address, work address, mailing address, and so forth).<br>The table must contain at least one row for each row in ACCOUNT_T. For example, you can have billing name and address information, technical contact name and address information, and sales name and address information. |
| **ACCOUNT_PHONES_T** | Contains a phone number and a phone type for each account. The table contains one row for each phone type (up to seven types are defined, including home, work, fax, and pager). If an account does not contain a phone number, the table does not contain a row for the account. |
| **BALANCE_GROUP_T** | Stores the balance information such as dollars, free minutes, bytes, and frequent flyer miles for various resources in an account. A balance group includes one or more sub-balances for each resource. The sub-balance contains the current amount, resource type, validity dates for the resource, rollover data, and sub-balance contributors. |
| **BAL_GRP_BALS_T** | Stores balance group data. |
| **BAL_GRP_SUB_BALS_T** | Stores sub-balance data. |
| **BILL_T** | Includes billing information, such as the amount due, amount adjusted, currency, and bill number. A **/bill** object is created at the end of a billing cycle.<br>A **/bill** object is created for every account. The account receivable for a bill is stored in the **/item** objects that point to the balance groups associated with the bill. The **/bill** object points to the **/account** object, the account's **/billinfo** object, and the **/invoice** object. |
| **BILLINFO_T** | Stores all billing, payment method, accounting cycle, and hierarchy information necessary to bill an account. A **/billinfo** object is created for every account. If the bill unit is subordinate, the **/billinfo** object points to the parent account and the parent account's **/billinfo** object. |
| **DEVICE_T** | Stores information about devices. The table contains a separate **/device** object for every device managed by BRM. Generic data applicable to all devices is stored in the parent **/device** object. Subclasses such as **/device/num** store information specific to a particular device type. |
| **DEVICE_NUM_T** | Stores device information specific to phone numbers managed by Number Manager. |
| **DEVICE_SERVICES_T** | Stores device/service mapping data. |
| **DEVICE_SIM_T** | Stores device information specific to SIM cards managed by SIM Manager. |

*Table 23–2 (Cont.) Tables Affected by the Conversion Process*

| Table Name | Description |
|---|---|
| **EVENT_T** | Contains a row for each account that has a nonzero balance forward amount. This table stores data for every event that occurs for an account or service. In the case of conversion of an account that has a balance, you should add a row to reflect the posting of that balance.<br>You also need to write an entry to this table for each entry you write to the EVENT_BILLING_DEAL_INFO_T and EVENT_BILLING_PRODUCT_ACTION_T tables.<br>In addition, you can optionally add rows to this table to store past activity for an account. For example, use this table to add rows for importing past billing and payment history into BRM. These rows can be brought over as memo type events, which can be viewed in Customer Center, but they do not affect the current account balance (which should be calculated and posted separately). |
| **EVENT_BAL_IMPACTS_T** | Stores event data. |
| **EVENT_ESSENTIALS_T** | Stores event data. |
| **GROUP_T** | Represents collections of other storable objects that have an assigned set of shared attributes. This table is optional. If you use it at conversion time, one row must be added to this table if the account is a parent or sponsoring account. |
| **GROUP_BILLING_ MEMBERS_T** | Rows for this table are optional and are included only for child accounts (that have a parent account) or for accounts that are members of a brand. During conversion, one row needs to be added to this table for each child account (regardless of the child account's payment method). |
| **GROUP_PERMITTEDS_T** | Indicates which accounts are group accounts. Rows for this table are optional and are included only for parent accounts that have at least one child account. |
| **GROUP_SHARING_ MEMBERS_T** | Stores members of a resource sharing group. |
| **GROUP_SHARING_ CHARGES_T** | Stores charges of a resource sharing group. |
| **GROUP_SHARING_ DISCOUNTS_T** | Stores discounts of a charge sharing group. |
| **GROUP_SHARING_ PROFILES_T** | Stores the profile data that is shared in a profile sharing group. |
| **ITEM_T** | Created to bundle events, this table summarizes billable item activity by type. Rows in this table are added for each row in the BILL_T table. In a conversion scenario, only one type of ITEM_T entry is important, the **/usage** item. Other items like the cycle forward item, payment item, dispute item, and so on are normally not pertinent for conversion. For conversion, a default **/item/misc** is created to bundle events. Also, a service-level item is created during conversion only if the recreate **='/item/<item-type>'** is specified in the input XML data. |
| **JOURNAL_T** | Stores general ledger (G/L) journal data. |
| **ORDERED_GROUPS_T** | Stores ordered balance group data. |
| **ORDERED_BALGROUP_T** | Stores ordered balance group data. |

*Table 23–2 (Cont.) Tables Affected by the Conversion Process*

| Table Name | Description |
|---|---|
| **PAYINFO_T** | Stores generic payment method information for an account. Only one row is added in this table for each row in ACCOUNT_T.<br>For each row in PAYINFO_T, you must add a row to the applicable payment method table. For example, add a row to PAYINFO_INV_T if the payment method for an account is **Invoice**.<br>Additionally, if you have created a custom payment method and a PAYINFO_*paymentMethod*_T table, where *paymentMethod* is the custom payment method, you must add a row to that table for each account with that payment method. |
| **PAYINFO_CC_T** | Contains a row for each account that has a payment method of **Credit Card**. This row is in addition to a row in PAYINFO_T.<br>**Note:** This payment method assumes that you are using the BRM-provided FirstUSAPaymentech or FDC modules to process your credit cards or that the clearing house or bank you use can process the same information. If your credit card processing is significantly different, you might want to set it up as another payment method and store information on that payment method separately. |
| **PAYINFO_DD_T** | Contains a row for each account that has a payment method of **Direct Debit**. This row is in addition to a row in PAYINFO_T. |
| **PAYINFO_INV_T** | Contains a row for each account that has a payment method of **Invoice**. Optionally, the table can also contain purchase order (PO) information. |
| **PROFILE_T** | Contains profile information, if any. This table can be left empty if you decide not to convert profile information. You can populate only one row in this table for each row in ACCOUNT_T. |
| **PROFILE_customTable_T** | Rows are added to this table only if rows are added to PROFILE_T, where *customTable* is the unique identifier you choose; for example, company name. For each row added to PROFILE_T, a corresponding row is added to PROFILE_**customTable**_T. |
| **PROFILE_ACCT_ EXTRATING_DATA_T** | Contains profile information. |
| **PROFILE_SERV_ EXTRATING_T** | Contains profile information. |
| **PROFILE_SERV_ EXTRATING_DATA_T** | Contains profile information. |
| **PURCHASED_DISCOUNT_T** | Contains an entry for each discount owned by an account at the time of conversion. |
| **PURCHASED_PRODUCT_T** | Contains an entry for each product owned by an account at the time of conversion. |
| **SERVICE_T** | Stores generic service type information for accounts. The table contains one row for each applicable service for each entry in ACCOUNT_T. In addition to a row in this table, a row must be created in the service type table, such as IP service or email. |

*Table 23–2 (Cont.) Tables Affected by the Conversion Process*

| Table Name | Description |
|---|---|
| **SERVICE_ALIAS_LIST_T** | Stores service aliases, used to identify customers by phone number. |
| **SERVICE_TELCO_T** | Stores telco service data. |
| **SERVICE_TELCO_ FEATURES_T** | Stores telco service data. |
| **SERVICE_TELCO_GSM_T** | Stores telco service data. |
| **UNIQUENESS_T** | Stores data that enforces uniqueness of logins across different database schemas in a multischema environment. |

## Audit Tables Affected by the Conversion Process

The following audit tables are affected by the conversion process:

- **AU_ACCOUNT_T**
- **AU_BAL_GRP_T**
- **AU_GROUP_SHARING_CHARGES_T**
- **AU_GROUP_SHARING_DISCOUNTS_T**
- **AU_GROUP_SHARING_PROFILES_T**
- **AU_GROUP_T**
- **AU_ORDERED_BALGROUP_T**
- **AU_ORDERED_GROUPS_T**
- **AU_PROFILE_ACCT_EXTRATING_DATA_T**
- **AU_PROFILE_SERV_EXTRATING_DATA_T**
- **AU_PROFILE_SERV_EXTRATING_T**
- **AU_PROFILE_T**
- **AU_PURCHASED_DISCOUNT_T**
- **AU_PURCHASED_PRODUCT_T**
- **AU_SERVICE_ALIAS_T**
- **AU_SERVICE_T**
- **AU_UNIQUENESS_T**

To configure Conversion Manager to load legacy data into other audit tables:

1. Go to the *BRM_home*/**apps/cmt/ctl_files** directory.
2. Copy the control file for the source table (*TableName*) to the *BRM_home*/**apps/cmt/ctl_files/audit** directory:

   ```
   cp TableName.ctl BRM_home/apps/cmt/ctl_files/audit/TableName.ctl
   ```

   where *TableName* is the name of the source table associated with the audit table. For example, to create a control file for the **AU_ACCOUNT_NAMEINFO_T** audit table, copy the control file for the **ACCOUNT_NAMEINFO_T** table to the *BRM_home*/**apps/cmt/ctl_files/audit** directory.
3. Open the *BRM_home*/**apps/cmt/ctl_files/audit/***TableName*.**ctl** file in a text editor.

4. Change all instances of *TableName* to **AU**_*TableName*.

   For example, change these lines:

   ```
   LOAD DATA
   APPEND
   INTO TABLE ACCOUNT_NAMEINFO_T
   ```

   to the following:

   ```
   LOAD DATA
   APPEND
   INTO TABLE AU_ACCOUNT_NAMEINFO_T
   ```

5. Save and close the file.
6. Open the *BRM_home*/**apps/cmt/ctl_files/Infranet.properties** file in a text editor.
7. Add *TableName* to the **infranet.cmt.auditloaders** line:

   ```
   infranet.cmt.auditloaders = account_t; bal_grp_t; group_sharing_charges_t;...;
   TableName
   ```

   For example, to add the control file for the **AU_ACCOUNT_NAMEINFO_T** table:

   ```
   infranet.cmt.auditloaders = account_t; bal_grp_t; group_sharing_charges_t;...;
   account_nameinfo_t;
   ```

8. Save and close the file.

# 24

# Migrating Data by Using New and Extended Storable Classes

This chapter describes how to migrate data to your Oracle Communications Billing and Revenue Management (BRM) database when the data requires new or extended storable classes.

For more information on migrating data, see "Understanding Conversion Manager".

**Important:** Conversion Manager is an optional feature that requires a separate license.

## About Migrating Data by Using New and Extended Storable Classes

If your legacy data includes data not currently supported in BRM, you support the custom data by creating storable classes or extending storable classes. You can then extend the Conversion Manager XML schema to migrate the data.

**Note:** Conversion Manager does not support extensions for all BRM classes. See "About Extended Storable Classes for Migration".

To use new or extended storable classes:

- Create the new or extended storable classes. See "About Extended Storable Classes for Migration".
- Create XML files that include the migrated data. See "Creating XML Files for New or Extended Storable Class Data".
- Create control files to add tables to the BRM database. See "Creating Control Files for Extended Storable Classes".
- Run the **pin_cmt** utility as you would for migrating non-extended data. See "Loading Legacy Data into the BRM Database".

**Important:**

- When you import data for new storable classes, you use a different **pin_cmt** utility parameter (**-import_custom**). When you migrate data into custom classes, the **pin_cmt** utility performs the additional step of activating the custom data when it is deployed.
- Import all custom data before you deploy any data. The only exception is if all the custom data belongs to the root account; for example, custom system configuration data.

## About Extended Storable Classes for Migration

You can extend the following storable classes:

- **/service**
- **/device**
- **/payinfo**
- **/profile**

For example:

- **/service/extension1**
- **/service/extension1/extension**

## About Linking to Data from New or Extended Storable Class Data

Data in new or extended storable classes can include links to other objects. For example, a **/service** extended object can include a link to an **/account** object by including the account ID in the PIN_FLD_ACCOUNT_OBJ field.

You can link to the following types of objects:

- **/account**
- **/service** (and extended **/service** storable classes)
- **/device** (and extended **/device** storable classes)
- **/payinfo** (and extended **/payinfo** storable classes)
- **/profile** (and extended **/profile** storable classes)
- **/group/sharing/charges**
- **/group/sharing/discounts**

When creating new or extended storable classes, follow the BRM standards and restrictions. For example:

- You cannot nest substructs.
- An array can include an array, but no further levels are allowed.

## Creating XML Files for New or Extended Storable Class Data

Use the following guidelines when creating an XML file for data for new or extended storable classes:

- Use the XML **myExtension** element for extended data.

- Use the XML **NewClass** element for new custom storable classes.
- Use the **myArray** and **mySubstruct** elements for arrays and substructs in the extended data.
- Use the **table** attribute in each array and substruct to define the table that stores the data.
- Use the **elem** attribute in each array to define the index.
- Use the **RefObj** element to create links to other objects. To do so, use the ID of the object being linked to. For example:

  ```
  <RefObj type ="/account"> 1234 </RefObj>
  ```

  If the **pin_cmt** utility does not find the object being linked to, it reports a warning and inserts a null value in the Portal object ID (POID) being referenced.
- Use the **type** attribute in all new and extended elements to define the type of data.
  - The **type** attribute for a *new* storable class is defined in the base storable class element. For example:

    ```
    <NewClass  id="1243" type="/myclass" table="myclass_t" read="L" write="L" >
    ```

  - The **type** attribute for an *extended* storable class is defined in the base storable class element. For example:

    ```
    <ActSrv id="771" type="/service/extended" precreate="/item/misc"
    global="true">
    ```

For XML examples, see the following sections:

- Example of Extending a Service Storable Class
- Example of Extending a Device Storable Class
- Example of Creating a Storable Class

### Creating an XSD File for Extended Data

To validate the XML file, you must create an XSD file for the extended information. You can use the **myExtension.xsd** file in *BRM_Home*/**apps/cmt/sample_data** as a starting point, where *BRM_Home* is the directory in which BRM is installed.

> **Important:** For each type of extended information, use a different namespace to avoid collisions caused by using the same name **myExtension**.

If the extended information contains BRM extensions (for example, **/service/***x*, **/payinfo/***x*, **/device/***x*, and **/profile/***x*), include the custom XSD in the **CMT_Subscribers.xsd** file. This allows the extended information to be validated along with the subscriber information.

If the extended information includes a custom storable class (for example, **/my_profile**), the custom XSD does not need to be included in the **CMT_Subscribers.xsd** file. The extended information can be validated by the custom XSD file.

Put the custom XSD file in the folder specified in the **infranet.cmt.schema.location** entry in the **pin_cmt** utility **Infranet.properties** file. By default, the folder is *BRM_Home*/**apps/cmt/schema_files/physical**.

**Important:** To enable XML validation, use the **infranet.cmt.preprocess.validation** entry in the **pin_cmt Infranet.properties** file. See "Enabling XML Validation".

## Creating Control Files for Extended Storable Classes

Control files add tables to the BRM database. You create a control file for each table. You need to create a control file for each of the following:

- New storable classes
- Substructs in new or extended storable classes
- Arrays in new or extended storable classes

When you create a control file, the name of the control file is used as the table name. For example, if you create a new storable class named **/myclass**, which includes a substruct named PIN_FIELD_MYSUBSTRUCT, the table name would be MYCLASS_SUBSTR_T. The control file name would be **myclass_substr_t.ctl**.

**Important:**

- The order of columns in the control file must be the same as the order of columns in the table.
- The order of data in the XML input file must be the same as the order of data in the control file and table.
- While adding tables to the BRM database, ensure that the tables listed in the **pin_cmt** utility **Infranet.properties** file belong to the main class.

For control file examples, see the following sections:

- Example of Extending a Service Storable Class
- Example of Extending a Device Storable Class
- Example of Creating a Storable Class

To use your custom control files:

1. Put the control files in *BRM_Home*/**apps/cmt/ctl_files**.
2. Edit the **pin_cmt** utility **Infranet.properties** file.
   - For control files that add substruct and array data, edit the **infranet.cmt.loaders** entry. Add the control file names. For example:

     ```
     infranet.cmt.loaders = account_t; account_nameinfo_t; ...; extended2_
     substr1_t; extended_cust1_t; extended2_cycle_t;
     ```

   - For control files that add data for a new storable class, add the **infranet.cmt.customtables** entry. Add the table names, separated by semicolons. For example:

     ```
     infranet.cmt.customtables = myclass_t;
     ```

     **Note:** By default, the **infranet.cmt.customtables** entry is not included in the **Infranet.properties** file.

### Creating Control Files for Custom Event Tables in a Virtual Column-Enabled System

Conversion Manager includes control files that are suitable for a virtual column-enabled system for migrating event tables (the **event_t** and **event_bal_impacts_t** tables). If you have custom event tables to be migrated (for event-type objects that are not in the BRM system), you must create the control files as described in Migrating Data by Using New and Extended Storable Classes. In addition, in a virtual column-enabled system, the Conversion Manager event control files cannot perform insert operations on the virtual columns (the *field_name*_**type** columns) of event tables in the BRM database; they must, instead, perform insert operations on the virtual column's associated supporting column *field_name*_**type_id**.

See *BRM System Administrator's Guide* for more information on using virtual columns on event tables.

## Example of Extending a Service Storable Class

The following example shows the XML data for an extended **/service** storable class:

```
<ActSub>
    <ActSrv id="771" type="/service/extended" precreate="/item/misc" global="true">
        <CWhn>2004-10-02T01:22:15-07:00</CWhn>
        <Log>159-51090219-237205-0-24083-1-tahoe</Log>
        <Pass>md5|5f4dcc3b5aa765d61d8327deb882cf99</Pass>
        <SrvSta>A</SrvSta>
        <Eff>2004-02-02T01:22:15-07:00</Eff>

        <SClsInfo>
        <myExtension>
            <mySubstruct table="extended_substr1_t" >
                <FldCustName type="string">abc</FldCustName>
                <FldCustType type="integer">99</FldCustType>
            </mySubstruct>
            <myArray table="extended_cust1_t" elem="1">
                    <FldAmt type="integer">190</FldAmt>
                    <FldDate="date">2004-02-02T01:22:15-7:00</FldDate>
                    <myArray table="extended_cycle_t" elem="1">
                        <Fldtype type="integer">190</Fldtype>
                    </myArray>
                </myArray>
        </myExtension>

        </SClsInfo>

    </ActSrv>
</ActSub>
```

In Developer Workshop, the extended **/service** storable class definition looks like this:

```
/service/extended
    CMT_ARRAY
        CMT_NAME
        PIN_FLD_ACCOUNT_TYPE
    CUST_ARRAY
        PIN_FLD_CYCLE
            PIN_FLD_CUSTOMER_TYPE
        PIN_FLD_CYCLE_DISC_AMT
        PIN_FLD_CYCLE_END_T
```

After loading the data, the extended fields are included in the stored object like this:

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /service/extended 11813 3
0 PIN_FLD_CREATED_T               TSTAMP [0] (1083337746) Fri Apr 30 08:09:06 2004
0 PIN_FLD_MOD_T                   TSTAMP [0] (1083338129) Fri Apr 30 08:15:29 2004
.
.
.
0 PIN_FLD_TYPE                    ENUM [0] 0
0 PIN_FLD_ALIAS_LIST              ARRAY [0] allocated 1, used 1
1     PIN_FLD_NAME                STR [0] "00491110102"
0 10000                           SUBSTRUCT [0] allocated 2, used 2
1     10004                       STR [0] "abc"
1     PIN_FLD_ACCOUNT_TYPE        ENUM [0] 99
0 10005                           ARRAY [1] allocated 3, used 3
1     PIN_FLD_CYCLE_DISC_AMT      DECIMAL [0] 190
1     PIN_FLD_CYCLE_END_T         TSTAMP [0] (1075713735) Mon Feb  2 01:22:15 2004
1     PIN_FLD_CYCLE               ARRAY [1] allocated 1, used 1
2         PIN_FLD_CUSTOMER_TYPE   ENUM [0] 190
```

The following examples show the control files used for loading the data for the extended **/service** storable class. The substruct and arrays require new tables. The array tables include the columns REC_ID or REC_ID2; the substruct does not include these columns.

### Example Control File for Substruct

```
-- extended_substr1_t.ctl

LOAD DATA
APPEND
    INTO TABLE EXTENDED_SUBSTR1_T
    (
        OBJ_ID0    INTEGER EXTERNAL TERMINATED BY ',',
        NAME       CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
        TYPE       INTEGER EXTERNAL TERMINATED BY ','
     )
```

### Example Control File for Array

```
-- extended_cust1_t.ctl

LOAD DATA
APPEND
    INTO TABLE EXTENDED_CUST1_T
    (
        OBJ_ID0     INTEGER EXTERNAL TERMINATED BY ',',
        REC_ID      INTEGER EXTERNAL TERMINATED BY ',',
        AMOUNT      INTEGER EXTERNAL TERMINATED BY ',',
        END_T       INTEGER EXTERNAL TERMINATED BY ','
     )
```

### Example Control File for Array

```
-- extended_cycle_t.ctl

LOAD DATA
APPEND
    INTO TABLE EXTENDED_CYCLE_T
    (
        OBJ_ID0    INTEGER EXTERNAL TERMINATED BY ',',
        REC_ID     INTEGER EXTERNAL TERMINATED BY ',',
```

```
        REC_ID2     INTEGER EXTERNAL TERMINATED BY ',',
        TYPE        INTEGER EXTERNAL TERMINATED BY ','
    )
```

### Setting a Service-Level Balance Group

The following example shows attributes in the input XML file that you set to associate a balance group with a service.

To create a balance group and associate it with a service, set the **bal_grp** attribute to **true**.

```
<ActSrv id="50001" type="/service/telco/gsm/telephony" precreate="/item/misc"
global="true" bal_grp="true" billInfoElem="1">
        <Log>159-20040202-210000-0-24083-1-zion</Log>
        <Pass>md5|5f4dcc3b5aa765d61d8327deb882cf99</Pass>
        <SrvSta>A</SrvSta>
        <CrtT>2004-05-26T01:22:15-07:00</CrtT>
        <Eff>2004-05-26T01:22:15-07:00</Eff>
        <ServId>ServiceTelephony</ServId>
        </ActSrv>
```

To associate the balance group of a **/billinfo** object with a service, set the **skip_bal_grp_crt** and **bal_grp** attributes to **true**. The balance group of the **/billinfo** object is associated with the service.

```
<ActSrv id="50001" type="/service/telco/gsm/telephony" precreate="/item/misc"
global="true" bal_grp="true" skip_bal_grp_crt="true" billInfoElem="1">
        <Log>159-20040202-210000-0-24083-1-zion</Log>
        <Pass>md5|5f4dcc3b5aa765d61d8327deb882cf99</Pass>
        <SrvSta>A</SrvSta>
        <CrtT>2004-05-26T01:22:15-07:00</CrtT>
        <Eff>2004-05-26T01:22:15-07:00</Eff>
        <ServId>ServiceTelephony</ServId>
        </ActSrv>
```

## Example of Extending a Device Storable Class

The following example shows the XML data for an extended **/device** storable class:

```
<DeviceInfo>
    <Device id="778" type="/device/extended2">
        <DevId>00982110309</DevId>
        <Src>source2</Src>
        <Mnufr>Airtel2</Mnufr>
        <Mdl>A4IR124</Mdl>
    </Device>
    <myExtension>
        <myArray table="dev_extended2_t" elem="1">
            <FldAmount type="integer">11</FldAmount>
            <FldDate type="date">2004-03-15T01:22:15-07:00</FldDate>
        </myArray>
    </myExtension>
</DeviceInfo>
```

In Developer Workshop, the extended **/device** storable class definition looks like this:

```
/device/extended2
    PIN_FLD_ENTRIES
        PIN_FLD_ENTRY_AMOUNT
        PIN_FLD_ENTRY_T
```

After loading the data, the extended fields are included in the stored object like this:

```
# number of field entries allocated 14, used 14
0 PIN_FLD_POID               POID [0] 0.0.0.1 /device/extended2 11811 0
0 PIN_FLD_CREATED_T          TSTAMP [0] (1083337746) Fri Apr 30 08:09:06 2004
0 PIN_FLD_MOD_T              TSTAMP [0] (0) <null>
0 PIN_FLD_READ_ACCESS        STR [0] "L"
0 PIN_FLD_WRITE_ACCESS       STR [0] "A"
0 PIN_FLD_ACCOUNT_OBJ        POID [0] 0.0.0.1 /account 1 0
0 PIN_FLD_DESCR              STR [0] ""
0 PIN_FLD_DEVICE_ID          STR [0] "00982110309"
0 PIN_FLD_MANUFACTURER       STR [0] "Airtel2"
0 PIN_FLD_MODEL              STR [0] "A4IR124"
0 PIN_FLD_SOURCE             STR [0] "source2"
0 PIN_FLD_STATE_ID           INT [0] 0
0 PIN_FLD_SERVICES           ARRAY [3] allocated 2, used 2
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 11806 0
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.1 /service/extended2 11813 0
0 PIN_FLD_ENTRIES            ARRAY [1] allocated 2, used 2
1     PIN_FLD_ENTRY_AMOUNT   DECIMAL [0] 11
1     PIN_FLD_ENTRY_T        TSTAMP [0] (1079342535) Mon Mar 15 01:22:15 2004
```

The following example shows the control file used for loading the data for the extended **/device** storable class:

```
LOAD DATA
APPEND
    INTO TABLE DEV_EXTENDED2_t
    (
        OBJ_ID0                 INTEGER EXTERNAL TERMINATED BY ',',
        REC_ID                  INTEGER EXTERNAL TERMINATED BY ',',
        AMOUNT                  INTEGER EXTERNAL TERMINATED BY ',',
        ENTRY_T                 INTEGER EXTERNAL TERMINATED BY ','
    )
```

## Example of Creating a Storable Class

The following example shows the XML data for a new storable class named **/myclass**:

```
<NewClass  id="1243" type="/myclass" table="myclass_t" read="L" write="L" >
    <AcctObj>0.0.0.1-indian500</AcctObj>
    <myArray table="myclass_rules_t" elem="0">
        <rum_id type="integer">222</rum_id>
        <descr type="string">sachin</descr>
        <start_t type="date">2006-10-02T01:22:15-07:00</start_t>
        <myArray table="myclassremit_t" elem="0">
            <remit_fld type="string">one1</remit_fld>
        </myArray>
        <myArray table="myclass_remit_t" elem="1">
            <remit_fld type="string">two1</remit_fld>
        </myArray>
        <myArray table="myclassremit_t" elem="2">
            <remit_fld type="string">three1</remit_fld>
        </myArray>
    </myArray>
</NewClass>
```

In Developer Workshop, the **/myclass** definition looks like this:

```
/myclass
```

```
                    PIN_FLD_ACCOUNT_OBJ
                    PIN_FLD_CREATED_T
                    PIN_FLD_MOD_T
                    PIN_FLD_POID
                    PIN_FLD_READ_ACCESS
                    PIN_FLD_RULES
                        PIN_FLD_REMIT_FLDS
                            PIN_FLD_REMIT_FLD
                        PIN_FLD_RUM_ID
                        PIN_FLD_SCENARIO_DESCR
                        PIN_FLD_SCHEDULE_DOWNTIME_START
                    PIN_FLD_WRITE_ACCESS
```

After loading the data, the stored object looks like this:

```
0 PIN_FLD_POID                          POID [0] 0.0.0.1 /myclass 43567 0
0 PIN_FLD_CREATED_T                     TSTAMP [0] (1083780256) 5/6/04 3:04 AM
0 PIN_FLD_MOD_T                         TSTAMP [0] (1083780256) 5/6/04 3:04 AM
0 PIN_FLD_READ_ACCESS                   STR [0] "L"
0 PIN_FLD_WRITE_ACCESS                  STR [0] "L"
0 PIN_FLD_ACCOUNT_OBJ                   POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_RULES                         ARRAY [0] allocated 4, used 4
1    PIN_FLD_RUM_ID                     INT [0] 222
1    PIN_FLD_SCENARIO_DESCR             STR [0] "sachin"
1    PIN_FLD_SCHEDULE_DOWNTIME_START    TSTAMP [0] (1159732335) 10/2/06 4:52 AM
1    PIN_FLD_REMIT_FLDS                 ARRAY [0] allocated 1, used 1
2        PIN_FLD_REMIT_FLD              STR [0] "one1"
1     PIN_FLD_REMIT_FLDS                ARRAY [1] allocated 1, used 1
2        PIN_FLD_REMIT_FLD              STR [0] "two1"
1    PIN_FLD_REMIT_FLDS                 ARRAY [2] allocated 1, used 1
2        PIN_FLD_REMIT_FLD              STR [0] "three1"
```

The following example shows the control file used for loading the data for the new **/myclass** storable class:

```
myclass_t.ctl

LOAD DATA
APPEND
    INTO TABLE MYCLASS_T
    (
        POID_DB             INTEGER EXTERNAL TERMINATED BY ',',
        POID_ID0            INTEGER EXTERNAL TERMINATED BY ',',
        POID_TYPE           CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
        POID_REV            INTEGER EXTERNAL TERMINATED BY ',',
        CREATED_T           INTEGER EXTERNAL TERMINATED BY ',',
        MOD_T               INTEGER EXTERNAL TERMINATED BY ',',
        READ_ACCESS         CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
        WRITE_ACCESS        CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
        ACCOUNT_OBJ_DB      INTEGER EXTERNAL TERMINATED BY ',',
        ACCOUNT_OBJ_ID0     INTEGER EXTERNAL TERMINATED BY ',',
        ACCOUNT_OBJ_TYPE    CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
        ACCOUNT_OBJ_REV     INTEGER EXTERNAL TERMINATED BY ','
    )
```

The following example shows the control file used for loading the data for the **/myclass** storable class PIN_FLD_RULES substruct:

```
myclass_rules_t.ctl

LOAD DATA
```

```
APPEND
    INTO TABLE MYCLASS_RULES_T
    (
        OBJ_ID0    INTEGER EXTERNAL TERMINATED BY ',',
        REC_ID     INTEGER EXTERNAL TERMINATED BY ',',
        RUM_ID     INTEGER EXTERNAL TERMINATED BY ',',
        DESCR     CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"',
        START_T   INTEGER EXTERNAL TERMINATED BY ','
     )
```

The following example shows the control file used for loading the data for the **/myclass** storable class PIN_FLD_REMIT_FLDS array:

```
myclass_remit_t.ctl

LOAD DATA
APPEND
    INTO TABLE MYCLASS_REMIT_T
    (
        OBJ_ID0     INTEGER EXTERNAL TERMINATED BY ',',
        REC_ID      INTEGER EXTERNAL TERMINATED BY ',',
        REC_ID2     INTEGER EXTERNAL TERMINATED BY ',',
        REMIT_FLD   CHAR TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
    )
```

## Example of Migrating Hierarchical Accounts

The following example shows the various attributes that you need to specify for different hierarchy-level accounts in the input XML file while performing the migration:

- **Parent account**: The following attributes should exist for the parent account:

  ```
  <ActSbsc id="0.0.0.1-10001" isParent="Y">
  ```

  Set **id** to the unique string by which the account will be identified. Set **isParent** to **Y**.

- **Paying child**: The following attributes should exist for the paying child account:

  ```
  <ActSbsc id="0.0.0.1-10002" parenRef="0.0.0.1-10001">
  ```

  Set **id** to the unique string by which the account will be identified. Set **parenRef** to a reference to the parent ID.

- **Nonpaying child**: The following attributes should exist for the nonpaying child account:

  ```
  <ActSbsc id="0.0.0.1-10003" parenRef="0.0.0.1-10001"
   payParenRef="0.0.0.1-10001">
  ```

  Set **id** to the unique string by which the account will be identified. Set **parenRef** to a reference to the parent ID. Set **payParenRef** to a reference to the paying parent ID.

  The **pay_type** (**PTyp**) attribute under **billinfo** should be set to **NPC** as follows:

  ```
  <ABinfo global="true">
              <ACDom>26</ACDom>
              <ANxt>2004-07-26T01:22:15-07:00</ANxt>
              <BlWn>1</BlWn>
              <BlSgmnt>RC</BlSgmnt>
  ```

```
            <PTyp>NPC</PTyp>
</ABinfo>
```

The **payinfo_obj_type** attribute under **payinfo** should be set to **/payinfo.** Also, comment out the **payinfo** section as follows:

```
<APinfo id="10004" type="/payinfo">
                    <DDom>26</DDom>
                    <RelDue>2004-05-26T01:22:15-07:00</RelDue>
                    <!-- payInvExtn>
                           <Add>340/C Baker Street</Add>
                           <Cty>Houston</Cty>
                           <Cntr>USA</Cntr>
                           <Nm>52345</Nm>
                           <Stt>Texas</Stt>
                           <Zip>56009</Zip>
                    </payInvExtn -->
</APinfo>
```

- **Paying and nonpaying child**: Child accounts can have multiple paying bill units and nonpaying bill units:
  - **Paying bill unit**: The following attributes should exist for each paying bill unit:

    ```
    <ActSbsc id="0.0.0.1-10004" parenRef="0.0.0.1-10001">
    <ActSrv id="0.0.1.1-10224" type="/service/telco/fixedretail"
    billInfoElem="2">
    <ABinfo elem="2" parentElem="1" bal_grp="False" payInfoRefId="0.0.0.1-114">
    <APinfo id="0.0.0.1-114" type="/payinfo/invoice">
    ```

  - **Nonpaying bill unit**: The following attributes should exist for each nonpaying bill unit:

    ```
    <ActSbsc id="0.0.0.1-10004" parenRef="0.0.0.1-10001">
    <ActSrv id="0.0.1.1-10334" type="/service/telco/prepaid" billInfoElem="1">
    <ABinfo elem="1" parentElem="1" payingParenRefId="0.0.0.1-10001">
    ```

    Set **payingParentRefId** to a reference to the parent account ID that owns the paying bill unit. This creates another **/billinfo** record for the child account. The **ar_billinfo_obj_id0** field of that record stores the ID of the paying bill unit.

## Setting an Account-Level Billinfo

The following example shows attributes in the input XML file that you set to define a **/billinfo** object as the account level **/billinfo**.

To define a **/billinfo** object as the account level **/billinfo**, set the **isAccBillinfo** attribute in the **<ABinfo> /billinfo** object to **Y**. The balance group that is associated with the **/billinfo** object then becomes the account-level balance group.

> **Note:** There can only be one **<ABinfo> /billinfo** object for which the **isAccBillinfo** attribute is set to **Y**. If there are duplicate **<ABinfo> /billinfo** objects, BRM returns an error and the migration stops.

```
<ABinfo global="true" spnrCnt="1" spnreeCnt="2" elem="1" bal_grp="true"
isAccBillinfo="Y" payInfoRefId="MyAPInfo_1">
       <ACDom>21</ACDom>
       <ActgType>B</ActgType>
```

```
        <ANxt>2013-07-01T01:22:15-07:00</ANxt>
        <CrtT>2013-06-07T01:22:15-07:00</CrtT>
        <BlWn>1</BlWn>
        <BlSgmnt>RC</BlSgmnt>
        <PTyp>INV</PTyp>
        <BISta>A</BISta>
        <BillStat>3</BillStat>
        <ExmtFrmColl>A</ExmtFrmColl>
        <BIStaFlag>SBCA</BIStaFlag>
        <BillInfoId>MyBillInfo_1</BillInfoId>
</ABinfo>
```

## Example of Migrating Hierarchical Bill Units within an Account

The following example shows hierarchical bill units within an account.

> **Note:** To create the hierarchical **/billinfo** objects within the same account, set the **thisActHierarchy** attribute to **true**. In the following example, the **/billinfo** objects BI1 and BI2 are in the account A. The BI1 **/billinfo** is the parent of BI2.

```
<ABinfo global="true" spnrCnt="1" spnreeCnt="2" elem="1" bal_grp="true"
payInfoRefId="MyAPInfo_1">
      <ACDom>26</ACDom>
      <ActgType>B</ActgType>
      <ANxt>2013-07-01T01:22:15-07:00</ANxt>
      <CrtT>2013-06-07T01:22:15-07:00</CrtT>
      <BlWn>1</BlWn>
      <BlSgmnt>RC</BlSgmnt>
      <PTyp>INV</PTyp>
      <BISta>A</BISta>
      <BillStat>3</BillStat>
      <ExmtFrmColl>A</ExmtFrmColl>
      <BIStaFlag>SBCA</BIStaFlag>
      <BillInfoId>MyBillId3</BillInfoId>
</ABinfo>
<ABinfo global="true" spnrCnt="1" spnreeCnt="2" elem="2" parentElem="1"
thisActHierarchy="true" bal_grp="true" payInfoRefId="MyAPInfo_1">
      <ACDom>26</ACDom>
      <ActgType>B</ActgType>
      <ANxt>2013-07-01T01:22:15-07:00</ANxt>
      <CrtT>2013-06-07T01:22:15-07:00</CrtT>
      <BlWn>1</BlWn>
      <BlSgmnt>RC</BlSgmnt>
      <PTyp>NPC</PTyp>
      <BISta>A</BISta>
      <BillStat>3</BillStat>
      <ExmtFrmColl>A</ExmtFrmColl>
      <BIStaFlag>SBCA</BIStaFlag>
      <BillInfoId>MyBillId1</BillInfoId>
</ABinfo>
<ABinfo global="true" spnrCnt="1" spnreeCnt="2" elem="3" parentElem="1"
thisActHierarchy="true" bal_grp="true" payInfoRefId="MyAPInfo_1">
      <ACDom>26</ACDom>
      <ActgType>B</ActgType>
      <ANxt>2013-07-01T01:22:15-07:00</ANxt>
      <CrtT>2013-06-07T01:22:15-07:00</CrtT>
      <BlWn>1</BlWn>
```

```
      <BlSgmnt>RC</BlSgmnt>
      <PTyp>NPC</PTyp>
      <BISta>A</BISta>
      <BillStat>3</BillStat>
      <ExmtFrmColl>A</ExmtFrmColl>
      <BIStaFlag>SBCA</BIStaFlag>
      <BillInfoId>MyBillId2</BillInfoId>
</ABinfo>
```

# 25

# Loading Legacy Data into the BRM Database

This chapter describes how to load legacy data into your Oracle Communications Billing and Revenue Management (BRM) database.

Before you can load data, you must extract it from the legacy database into XML files. See the following chapters:

- Understanding Conversion Manager
- Mapping Legacy Data to the BRM Database

For more information on performance tuning, see "Improving Conversion Manager Performance".

> **Important:** Conversion Manager is an optional component, not part of base BRM, and requires a separate license.

## Importing Data

You import legacy data into the BRM database one file at a time.

> **Important:** To specify a database connection, edit the **pin_cmt utility Infranet.properties** file in *BRM_Home*/**apps/cmt**. *BRM_Home* is the directory in which you installed the BRM software.

To import legacy data into the BRM database:

1. Go to *BRM_Home*/**apps/cmt**.
2. Do one of the following:
   - To import data that is not stored in a new class, run the **pin_cmt** utility using the following syntax:

     ```
     pin_cmt -import -file XML_input_data_file stage_ID
     ```

     For example:

     ```
     pin_cmt -import -file data.xml 100
     ```

     > **Important:** If you have a multischema system, make sure the stage IDs are larger than the database schema IDs. For example, if you have a schema with the number 0.0.0.5, use stage IDs larger than 5.

- To import data that is stored in a new class, run the **pin_cmt** utility using the following syntax:

  ```
  pin_cmt -import_custom -file XML_input_data_file stage_ID
  ```

  For example:

  ```
  pin_cmt -import_custom -file data.xml 100
  ```

See "pin_cmt" for more information.

Possible errors:

- On all parser errors, Conversion Manager rejects the whole file. Take corrective action based on the errors logged in **cmt.pinlog** and resubmit the corrected file.
- A record is rejected and not imported if its reference object is not found. For example, in case of importing a child account when the parent account is not found, the child record is rejected. The error is noted in the log file.
- I/O errors, such as the inability to find or open the specified document. The whole file is rejected and an error is logged.
- You run out of database space. Use **pin_cmt** with the **-recover** parameter to recover your data. For more information, see "Reloading Data".

## Deploying Converted Data

When you deploy the data, the staging accounts are made available for production by updating the database ID number in the object Portal object IDs (POID).

**Important:** To specify a database connection, edit the **pin_cmt utility Infranet.properties** file in *BRM_Home*/**apps/cmt**.

To deploy your converted data, run **pin_cmt** using the following syntax:

```
pin_cmt -deploy DOM stage_ID
```

where:

- *DOM* is the billing cycle's day of month.
- *stage_ID* is the identity of the staging area.

See "pin_cmt" for more information.

## Reloading Data

If **pin_cmt** runs out of space in your BRM database for data rows, the importing process stops. Data that was not imported can be imported after more space is made available in the database.

**Important:** To specify a database connection, edit the **pin_cmt** utility **Infranet.properties** file in *BRM_Home*/**apps/cmt**.

To import data when the utility runs out of database space:

1. Add space to the database.

2. Read the log files in *BRM_Home*/**apps/cmt** to find the following information:
   - How many records were processed.
   - The batch ID for the import process that did not complete.
3. Edit the control files:
   a. At the beginning of every control file, replace the string LOAD_DATA with CONTINUE_LOAD_DATA.
   b. In each table's control file, specify the number of records to skip for that table by using the INTO TABLE clause. For example:

      ```
      INTO TABLE account_t
      SKIP 756
      ```

      where **756** is the number of previously processed records.
4. Run the **pin_cmt** utility with the **-recovery load** parameter:

   **pin_cmt -recovery load** *batch_ID*

   See "pin_cmt" for more information.

# Troubleshooting Conversion Manager

When Conversion Manager imports legacy data into the BRM database, it creates a log file (**cmt.pinlog**) with a list of errors and warnings depending on the reporting level set for message logging. (See "Setting the Reporting Level for Logging Messages" in *BRM System Administrator's Guide*.)

In addition, the log file lists successfully processed records and failed records.

To find any error messages, read the **cmt.pinlog** file in the *BRM_Home*/**apps/cmt** directory.

You can also read the Connection Manager and Data Manager log files.

## Common pin_cmt Utility Error Messages

Table 25–1 shows common **pin_cmt** error messages.

***Table 25–1 Common pin_cmt Messages***

| Error Message | Description |
|---|---|
| BAD_INFRANET_CONNECTION | One of the following:<br>■ BRM is not running<br>■ The CM connection information in the **pin_cmt** utility **Infranet.properties** file is incorrect. |
| CMD_LINE_ARG_ERR | Error in the **pin_cmt** utility command line syntax. |
| DB_CONNECTION_ERR | The database connection configuration in the **pin_cmt** utility **Infranet.properties** file is incorrect. This error occurs when the database is down. |
| DUPLICATE_ACC_BILLINFO_ERR | An account in the input XML file contains more than one **/billinfo** object (**<ABinfo>**) with **isAccBillinfo** set to **Y**. |
| DUPLICATE_ELEM_SERVICEALIASLIST | The XML input file contains a duplicate elementary value in the service alias lists within a service. |

*Table 25–1 (Cont.) Common pin_cmt Messages*

| Error Message | Description |
|---|---|
| FILE_NOT_FOUND_ERR | The input XML file is missing from the location specified in the command line. |
| MISSING_RESOURCE_ERR | A required configuration entry in the **pin_cmt** utility **Infranet.properties** file is missing. |
| PARSING_ERR | The input XML file is either not well-formed or not valid with respect to CMT XSD. |
| IL_PR_PARENT_NOT_FOUND_ERR | The input XML file includes an incorrect parent (**/group/billing**) reference. |
| IL_PR_PAYING_PARENT_NOT_FOUND_ERR | The input XML file includes an incorrect paying parent (**/group/billing**) reference. |
| INCORRECT_DEVICE_REF | The input XML file includes an incorrect device reference. |
| INCORRECT_SUB_OBJ_SERVICE_REF | The input XML file includes an incorrect subscription service reference. |
| INCORRECT_GSC_PARENT_REF | The input XML file includes an incorrect group sharing charges reference. |
| INCORRECT_GSD_PARENT_REF | The input XML file includes an incorrect group sharing discounts reference. |
| INCORRECT_GSP_PARENT_REF | The input XML file includes an incorrect group sharing profiles reference. |
| PROCESS_IS_RUNNING | The input XML file is either already loaded or currently being loaded by another **pin_cmt** instance. |
| SQL_ERROR | Internal **pin_cmt** error. |

# Testing the Imported Data

You test the data to verify that the storable objects have been created correctly and that record pointers are consistent.

Use any of the following tools to validate the data in your BRM database:

- The **testnap** utility and Object Browser. See "Using testnap and Object Browser to Validate the Database".
- Customer Center. See "Using Customer Center to Validate Data".
- SQL. See "Using SQL to Validate Data".

> **Tip:** Before performing the initial test conversion, prepare a test database by using the **pin_setup** script, and then create a backup of the database. Then load the current price plan and create another backup. This saves time because it is easier to reload a database than create another one.

## Using testnap and Object Browser to Validate the Database

Use **testnap** to verify that the following storable objects have been created correctly in the BRM database. Use Object Browser to look at the contents of each new object in the BRM database.

- **/account**

- **/bill**
- **/item**
- **/event**
- **/service**
- **/group**
- **/payinfo**
- **/billinfo**
- **/balance_group**
- **/device**
- **/device/num**
- **/device/sim**

> **Tip:** Print the results of the your **testnap** commands and match the objects in the list. This allows you to make any necessary notes.

For more information on how to use **testnap**, see "Using testnap" in *BRM Developer's Guide*.

### Validating /account Objects

To validate **/account** objects:

1. Use **testnap** or Object Browser to display the data in the object.
2. Examine each field to ensure the data in the field matches the input data file.
3. Verify that all the products and services owned by this account are present.
4. Verify that all of the balances for this account are correct.

### Validating /bill, /item, /event, /service, and /payinfo Objects

To validate these objects:

1. Find the POIDs of these objects in the **/account** object.
2. Use **testnap** or Object Browser to display the data for each of these objects.
3. Examine each field in the object to ensure that the data contained in the field matches the data in the input data file.
4. Ensure that these objects reference the correct **/account** object.

## Using Customer Center to Validate Data

To validate data, verify the following:

- You can retrieve account data without any errors.
- You can update an account without any errors.
- You can change payment methods successfully. For example, change an account payment method from credit card to invoice and back to credit card (use the **answer_s** and **answer_b** daemons, if necessary).

- If parent-child billing data was converted, verify that the parent-child grouping works correctly. To check this, do the following:
  - Change some of the existing child accounts to orphan accounts, and some of the existing orphan accounts to child accounts.
  - Add a few arbitrary hierarchies.
  - Move accounts to and from the parent accounts and verify that there are no errors.

## Using SQL to Validate Data

You can use SQL to check the various record counts in BRM.

> **Note:** BRM contains the root account, which increases the number of accounts by 1. Remember to take this into account at the time of validation.

Verify the following:

- The total number of accounts created is equal to the total number of accounts converted from your legacy system. Use the following SQL statement:

  ```
  select count(*) from ACCOUNT_T
  ```

- The total number of account name and address records is equal to the total number of accounts converted from your legacy system. Use the following SQL statement:

  ```
  select count(*) from ACCOUNT_NAMEINFO_T
  ```

  > **Note:** If your implementation has more than one name and address type, you need to take this into account.

- The total number of bill objects is equal to the total number of accounts converted from your legacy system. This number must equal the number of records in the ACCOUNT_T table. Use the following SQL statement:

  ```
  select count(*) from BILL_T
  ```

- The total number of payment objects is equal to the total number of accounts converted from your legacy system. This number should equal the number of records in the ACCOUNT_T table. Use the following SQL statement:

  ```
  select count(*) from PAYINFO_T
  ```

  Also verify that the total records in the PAYINFO_INV_T, PAYINFO_CC_T, and PAYINFO_DD_T tables match the number of records in the ACCOUNT_T table. Use these SQL statements:

  ```
  select count(*) from PAYINFO_INV_T
  select count(*) from PAYINFO_CC_T
  ```

- The total number of **/profile** objects (if created) is equal to the total number of accounts converted from your legacy system: this number should equal the number of records in the ACCOUNT_T table. Use the following SQL statement:

  ```
  select count(*) from PROFILE_custom_table_T
  ```

where *custom_table* is the implementation-unique identifier you choose; for example, company name.

- The total number of subordinate (child) accounts match the number of rows in the GROUP_BILLING_MEMBERS_T table. Use the following SQL statement:

```
select count(*) from GROUP_BILLING_MEMBERS_T where
object_type = '/account'
```

- The total number of parent accounts match the number of rows in the GROUP_T table. Use the following SQL statement:

```
select count(*) from GROUP_T where poid_type= '/group/billing'
```

- The total number of parent accounts match the number of rows in the GROUP_PERMITTEDS_T table. Use the following SQL statement:

```
select count(*) from GROUP_PERMITTEDS_T where type = '/account'
```

# 26

# Migrating Legacy Data into BRM Table Partitions

This chapter describes how to load data from legacy billing systems into Oracle Communications Billing and Revenue Management (BRM) table partitions.

Before you read this chapter, you should be familiar with the following:

- Partitioning. See "Partitioning Tables" in *BRM System Administrator's Guide*.
- Conversion Manager. See "Understanding Conversion Manager".

## About Migrating Legacy Data into Table Partitions

To load legacy data into BRM table partitions, do the following:

- Create back-dated partitions for storing legacy data. See "About Your Partitioning Scheme".
- Configure Conversion Manager to encode a creation timestamp into specified object POIDs. See "About Making Conversion Manager Aware of Partitions".

## About Partitioning

Some BRM tables, such as the BILL_T table, store large amounts of data, which can slow BRM search times. Dividing large tables into smaller, more manageable chunks, called partitions, makes it easier for BRM to find data because it needs to search only a single partition for data rather than an entire table.

In BRM, you create table partitions based on dates and divided by a specified interval: month, week, or day. For example, you could set one partition for May 1 to May 31, the subsequent one for June 1 to June 30, and so on. Objects are stored in partitions according to their creation date and time. For example, **/event** objects created in May 2009 could be stored in the May 2009 partition of the EVENT_T table.

### About Your Partitioning Scheme

Before you begin loading legacy data into BRM, you must determine your partitioning scheme, including which tables to partition and how many months of legacy data to load into BRM. For example, you might decide to load only the last five months of your billing data into the BRM system. In this case, you would create five back-dated partitions with a monthly interval for the BILL_T table.

After determining your partitioning scheme, you must create the following:

- **Back-dated table partitions for your legacy data**. For example, if you are loading legacy billing and item data into BRM, create back-dated table partitions for the **/bill** storable class (BILL_T table) and the **/item** storable class (ITEM_T table). Your back-dated tables must also comply with the following restrictions:
  - The oldest back-dated partition that you create must have a start date prior to the creation date of the oldest legacy object you are loading into BRM. For example, if the oldest object that you are loading has a creation date of March 8, 2005, you must create a back-dated table partition that starts on or before March 7, 2005.
  - The most recent back-dated partition that you create must have an end date that is at least one day prior to the BRM installation date. For example, if you install BRM on August 15, 2009, the most recent back-dated partition must end on or before August 14, 2009.
- **Future-dated partitions for data created on your new BRM system**. Create future-dated partitions for any storable classes you would like partitioned. The first future-dated partition must start two days after the BRM installation date. For example, if you install BRM on August 15, 2009, the first partition's start date must be August 17, 2009, or later.

For example, assume you install BRM on August 15, 2009, and the oldest billing object in your legacy system has a creation date of January 20, 2009. In this example, you would create seven back-dated partitions and one or more future-dated partitions for your **/bill** storable class (BILL_T table) as shown in Table 26–1:

***Table 26–1 Partitioning Scheme Example***

| Partition | Date Range for Each BILL_T Partition |
|---|---|
| 1 | January 15, 2009, through February 14, 2009 |
| 2 | February 15, 2009, through March 14, 2009 |
| 3 | March 15, 2009, through April 14, 2009 |
| 4 | April 15, 2009, through May 14, 2009 |
| 5 | May 15, 2009, through June 14, 2009 |
| 6 | June 15, 2009, through July 14, 2009 |
| 7 | July 15, 2009, through August 14, 2009 |
| 8 | August 17, 2009, through September 16, 2009 |
| 9 | September 17, 2009, through October 16, 2009 |
| 10 | **partition_last** |

In this example, objects are loaded into partitions as shown below:

- Objects with creation timestamps prior to February 15, 2009, are automatically loaded into Partition 1.
- Objects with creation timestamps on or after February 15, 2009, and prior to March 15, 2009, are loaded into Partition 2.
- Objects with creation timestamps after August 14, 2009, and prior to August 17, 2009, are loaded into Partition 8.

**Note:** If an object's creation timestamp is not covered by one of the partition ranges, the object is automatically loaded into the next higher-range partition.

- Objects with creation timestamps after October 16, 2009, are loaded into partition 10 (**partition_last**).

**Note:** If an object's creation timestamp is not covered by one of the partition ranges and a higher-range partition does not exist, the object is automatically loaded into **partition_last**.

# About Making Conversion Manager Aware of Partitions

Conversion Manager can now encode timestamps into the object POIDs of partitioned and purgeable storable classes, allowing the object to be loaded into the correct table partition. For example, if a legacy bill object was created in May, Conversion Manager encodes the creation timestamp into the **/bill** object's POID. The **/bill** object can then be loaded into the BILL_T table's May 2009 partition.

## About the Timestamps Encoded in Object POIDs

The timestamp encoded in object POIDs can be either:

- The creation timestamp passed in by the legacy system. In this case, you must pass the object creation timestamp in the **CrtT** element of the input XML file.
- The system time when you run Conversion Manager. It does not use the timestamp from **pin_virtual_time**.

You must make sure that the creation time is applied consistently to all related objects. For example:

- When you migrate unpaid bills and open items from your legacy billing system to BRM, the bill objects and their associated item objects must all use the legacy object's creation time. Otherwise, the billing and invoicing utilities will not find all items associated with a bill.
- When you migrate balances from your legacy system to BRM, the account objects and their associated balances must all use the legacy object's creation time.

## Conversion Manager Tasks for Partitioned Storable Classes

When Conversion Manager is configured to encode timestamps into object POIDs, it performs these additional tasks:

- Determines which storable classes are partitioned and purgeable by reading the BRM data dictionary. In the data dictionary, partitioned and purgeable storable classes have their IS_PARTITIONED field set to **1** and their PARTITION_MODE field set to **Finite**.
- Reserves a set of POIDs for each partitioned, purgeable storable class and a single set of POIDS for all nonpartitioned storable classes. For example, if the **/bill** and **/item** storable classes are partitioned and purgeable, Conversion Manager reserves three sets of POIDs: one for **/bill** objects, one for **/item** objects, and one for objects of all nonpartitioned storable classes.
- Assigns POIDs to objects based on the storable class type:

- If the storable class is partitioned and purgeable, Conversion Manager assigns a POID from the storable class's reserve of partitioned POIDs and encodes the timestamp. It uses either the timestamp passed in from the legacy system or the system time when you ran Conversion Manager.
- If the storable class is not partitioned, Conversion Manager assigns a POID from the reserve of nonpartitioned POIDs.

**Note:** If the set of reserved POIDs is already depleted, Conversion Manager reserves a new set of POIDs before assigning one to an object.

The object is then loaded into a partition according to the storable class type and the timestamp encoded in the object POID.

During the loading phase, if no corresponding partition is available for the timestamp encoded in the object POID, the object is loaded into the next higher-range partition. If no higher-range partition is available, the object is loaded into **partition_last**.

If a storable class *is not* partitioned, the object is loaded directly into the target table.

### About Configuring Conversion Manager to Encode Timestamps in POIDs

To configure Conversion Manager to encode timestamps in object POIDs, you specify the following in the Conversion Manager **Infranet.properties** file:

- The number of POIDs to reserve at a time.
- Which timestamp to encode into POIDs.

For more information, see "Configuring Conversion Manager for Partitioning".

## Setting Up Your System to Load Legacy Data into Table Partitions

To set up your system to load data from legacy billing systems into BRM table partitions:

1. Make sure partitioning is enabled in your BRM system by doing either or both of the following:
   - When installing BRM, choose to partition your tables.
   - After installing BRM, convert nonpartitioned storable classes to partitioned storable classes.

   For more information, see "Partitioning Tables" in *BRM System Administrator's Guide*.
2. Determine the oldest object for each storable class that will be partitioned in the BRM database. The oldest partition's start date must be prior to the oldest object's creation date for that storable class.
3. Create partitions with the **partition_utils** utility. See "Creating Partitions for Your Legacy Data".
4. Configure Conversion Manager for partitioning. See "Configuring Conversion Manager for Partitioning".
5. Include the legacy object's creation timestamp in the Conversion Manager input XML file. See "Passing Object Creation Timestamps in the Input XML File".

## Creating Partitions for Your Legacy Data

To create back-dated partitions, enter the following command:

```
partition_utils  -o add  -t realtime  -s start_date  -u month|week|day  -q quantity
[-c storable_class]  [-w width]  -b
```

where:

- *start_date* specifies the starting date for the first partition. The format is *MMDDYYYY*. The start date must be prior to the BRM installation date. The oldest partition's start date must occur prior to the oldest object's creation date.
- *quantity* specifies the number of partitions to add.
- *storable_class* specifies the storable class to store in the partition. The default is **/event**.
- *width* specifies the number of units in a partition (for example, 3).

You must also create future-dated partitions for objects generated by your new BRM system. For more information on how to create future-dated partitions, see "Partitioning Tables" in *BRM System Administrator's Guide*.

## Configuring Conversion Manager for Partitioning

By default, Conversion Manager does not encode timestamps in object POIDs. You configure Conversion Manager to do so through its **Infranet.properties** file.

To configure Conversion Manager for partitioning:

1. Open the *BRM_Home*/**apps/CMT/Infranet.properties** file in a text editor. *BRM_Home* is the directory in which you installed the BRM software.
2. Set the entries for creating partitioning-aware object POIDs:
   - Use **infranet.cmt.timestampvalidation** to specify which timestamp to encode in generated POIDs. See "Configuring which Timestamp to Encode".
   - Use **infranet.cmt.noofrecords** to specify the number of POIDs to reserve for partitioned POIDs. See "Configuring the Number of POIDs to Reserve".
3. Save and close the file.

### Configuring which Timestamp to Encode

The timestamp encoded in object POIDs can be either the object creation time passed in from the legacy system or the system time when you run **pin_cmt**.

> **Important:** If you specify to use the system time, Conversion Manager loads all of the legacy data into one partition. You must configure and size the appropriate partition to store the volume of data that will be loaded into it.

You specify which timestamp to encode by using the following **Infranet.properties** file entry:

```
infranet.cmt.timestampvalidation = Value
```

where *Value* is one of the following:

- **0**: The timestamp is set to the creation time passed in the input XML file. If a creation time is not passed in, Conversion Manager assigns the system time when you run **pin_cmt**.
- **1**: The timestamp is set to the creation time passed in the XML file only. If a creation time *is not* passed in, Conversion Manager generates an error. This setting ensures that all partitioned classes include the original creation timestamp. This is the default.
- **2**: The timestamp is set to the system time when you run **pin_cmt**. If a creation timestamp *is* passed in, Conversion Manager generates an error. This ensures that all partitioned classes are loaded with the system time.

> **Caution:** This parameter cannot enforce validations across multiple runs of Conversion Manager. Loading some objects with the system time and other objects with the object creation time could cause system inconsistencies and business operations to fail.

### Configuring the Number of POIDs to Reserve

Conversion Manager reserves a set of POIDs for each storable class type that is partitioned and reserves another set of POIDs for all other nonpartitioned storable classes.

To configure the number of POIDs to reserve:

- For partitioned storable classes, use the following entry:

  ```
  infranet.cmt.noofrecords = Number
  ```

  where *Number* specifies the number of POIDs to reserve. The default is **1**.

  Conversion Manager reserves the specified number of POIDs for each storable class that is partitioned.

- For nonpartitioned storable classes, use the following entries:

  ```
  infranet.cmt.noofrecords = Number
  infranet.cmt.avgnoofservices = Services
  infranet.cmt.avgnoofdevices = Devices
  ```

  where:

  - *Number* specifies the number of POIDs to reserve. The default is **1**.
  - *Services* specifies the average number of services in your accounts. The default is **5**.
  - *Devices* specifies the average number of devices in your accounts. The default is **2**.

  Conversion Manager multiplies these three values (*Number* * *Services* * *Devices*) to compute the total number of POIDs to reserve for nonpartitioned storable classes.

For example, assume you partition the **/bill** and **/item** storable classes and the **Infranet.properties** file includes the following entries:

```
infranet.cmt.noofrecords = 1000
infranet.cmt.avgnoofservices = 5
infranet.cmt.avgnoofdevices = 3
```

In this example, Conversion Manager reserves the following POIDs:

- 1,000 POIDs for **/bill** objects.
- 1,000 POIDs for **/item** objects.
- 15,000 POIDs for all other objects.

## Passing Object Creation Timestamps in the Input XML File

The Conversion Manager input XML file includes a **CrtT** element for all storable classes. If you want to encode the legacy object's creation time in your POIDs, you must pass the object's creation time in the **CrtT** element of the input XML file.

When passing the legacy object's creation time, make sure:

- The object creation timestamp is correct. Conversion Manager does not perform any validations on values passed in the **CrtT** element.
- Objects using the legacy creation time are not combined in the same partitioned, purgeable table as objects using the system time.

# Part IV

## Customer Management Utilities

Part IV provides reference information about customer management utilities. It contains the following chapters:

# 27

# Conversion Manager Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Conversion Manager utilities.

# cmt_mta_cycle_fees

**Important:** The **cmt_mta_cycle_fees** utility is run internally by the **pin_cmt** utility. Do not run this utility by itself.

The **cmt_mta_cycle_fees** utility applies cycle forward fees to BRM accounts that have been deployed by the **pin_cmt** utility. See "Understanding Conversion Manager" and "Loading Legacy Data into the BRM Database".

**Important:** Conversion Manager is an optional component, not part of base BRM.

**Note:** To connect to the BRM database, the **cmt_mta_cycle_fees** utility needs a configuration file in the directory from which you run the utility.

## Location

*BRM_Home*/**apps/cmt**

where, *BRM_Home* is the directory in which you installed the BRM software.

## Syntax

```
cmt_mta_cycle_fees -stage_id stage_ID -cycle_dom day_of_month
```

## Parameters

**-stage_id *stage_ID***
The stage ID used when the data was imported.

**-cycle_dom *day_of_month***
The billing day of month for the accounts that need cycle fees applied.

## Results

The **cmt_mta_cycle_fees** utility notifies you when it runs successfully. Otherwise, look in the **pin_mta.pinlog** file in *BRM_Home*/**apps/cmt** for errors.

# pin_cmt

Use the **pin_cmt** utility to load legacy data in XML format into your BRM database. See "Understanding Conversion Manager" and "Loading Legacy Data into the BRM Database".

**Important:** Conversion Manager is an optional component, not part of base BRM.

**Note:** To specify a database connection, edit the **pin_cmt Infranet.properties** file in *BRM_Home*/**apps/cmt**.

## Location

*BRM_Home*/**apps/cmt**

## Syntax

```
pin_cmt -import -file XML_input_data_file stage_ID|
        -import_custom -file XML_input_data_file stage_ID|
        -recovery load batch_ID|
        -deploy DOM stage_ID
```

## Parameters

**-import -file *XML_input_data_file stage_ID***
Imports the specified data file into the BRM database.

**Note:** Ensure that there are no extra spaces in the input XML file. If you need to indent text in the XML file, use the TAB key to add space.

**-import_custom -file *XML_input_data_file stage_ID***
Imports data that uses new storable classes into the BRM database.

**-recovery load *batch_id***
Reloads data after a failed load process. The batch ID is recorded in the **cmt.pinlog** file in *BRM_Home*/**apps/cmt.**

**-deploy *DOM stage_ID***
Deploys data. DOM is the billing cycle's day of the month. *stage_ID* is the stage ID used when the data was imported.

In this example, accounts with a billing day of month of 15 and stage ID of 100 are deployed:

```
pin_cmt -deploy 15 100
```

## Results

The **pin_cmt** utility notifies you when it runs successfully. Otherwise, look in the **cmt.pinlog** file in *BRM_Home*/**apps/cmt** for errors. In addition, look for errors in the log file specified in the **pin.conf** file for the **cmt_mta_cycle_fees** utility.

# 28

# Customer Management Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) customer management utilities.

# load_notification_event

Loads notification XML file from Pipeline Manager into the BRM database. This allows BRM to notify customers when their balance has reached a threshold value during the batch rating process. For more information, see "About Credit Limit and Threshold Checking during Batch Rating".

You must configure the Batch Controller to execute this utility. See "Configuring Batch Controller to Run load_notification_event".

**Note:** To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## Location

*BRM_Home*/**bin**

where, *BRM_Home* is the directory in which you installed the BRM software.

## Syntax

```
load_notification_event [-d] [-v] [-h] XML_file
```

## Parameters

**-d**
Sets the log level to debug and outputs debug information into the log file for this process. If not set, only error-level information is output.

**-v**
Displays information about failed or successful processing as the utility runs.

**-h**
Displays syntax and parameters for this utility.

***XML_file***
The name and location of the XML file to load into the BRM database. This must be the last parameter listed on the command line.

## Results

This utility notifies you when it successfully loads the XML file.

If the utility does not notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

# load_pin_business_profile

Use this utility to load business profiles and validation templates into the BRM database.

You enter this information in the business profile configuration file (*BRM_Home*/**sys/data/config/pin_business_profile.xml**).

For more information, see the following topics:

- About Business Profiles
- Setting Up Business Profiles and Validation Templates
- Editing the Business Profile Configuration File

**Note:** This utility supports branding.

**Important:** To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. For more information on creating configuration files for BRM utilities, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## 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 (see "Validating Your Business Profile Configuration File Edits"). This option does *not* create, modify, or delete any business profile or validation template objects in your BRM database.

**Tip:** 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. See "Validating Your Business Profile Configuration File Edits".

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.

To verify that the business profile and validation template information was loaded, display the **/config/business_profile** and **/config/template** subclass objects by using one of the following features:

- Object Browser
- **robj** command with the **testnap** utility

For more information on 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*.

> **Important:** You must stop and restart the Connection Manager (CM) to make new business profile and validation template data available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

# load_pin_customer_segment

Use this utility to load your customer segment definitions into the BRM database. You define customer segments in the *BRM_Home***/sys/data/config/pin_customer_segment.xml** file.

For more information on customer segments, see "Creating and Managing Customer Segments".

**Note:** You cannot load separate **/config/customer_segment** objects for each brand. All brands use the same object.

**Caution:** When you run the **load_pin_customer_segment** utility, it overwrites the existing customer segments in the database. If you are updating your customer segments, you cannot load new segments only. You must load complete sets of customer segments each time you run the **load_pin_customer_segment** utility.

**Important:** To connect to the BRM database, the **load_pin_customer_segment** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## 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.

To verify that the customer segments were loaded, you can display the **/config/customer_segment** object by using the *Object Browser*, or use the **robj** command with the **testnap** utility. See "Reading an Object and Writing its Contents to a File" in *BRM Developer's Guide*.

**Important:** You must restart the Connection Manager to start recording your customer login failure preferences. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

# 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*.

**Note:** You cannot load separate **/config/notify** objects for each brand. All brands use the same object.

**Caution:** If your database already contains an event notification list when you run this utility, the utility replaces the list in the database with the list in the configuration file that you load. If you use event notification for multiple features, you must merge the old list with the new list *before* running this utility. Otherwise, you will lose existing event notification functionality. See "Merging Event Notification Lists" in *BRM Developer's Guide*.

**Important:** To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator'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*.

> **Tip:** 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** storable 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.

> **Important:** You must restart the Connection Manager to make new events and the corresponding opcodes available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

# load_pin_verify

Use this utility to load your preferences for recording customer login failures into the BRM database. You define preferences for recording customer login failures in the *BRM_Home***/sys/data/config/pin_verify** file.

For more information on monitoring customer login failures, see "Tracking Customer Authentication and Authorization Records".

**Note:** You cannot load separate **/config/verify** objects for each brand. All brands use the same object.

**Caution:** When you run this utility, 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 utility.

**Important:** To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## 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**.

**Tip:** 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.

**Important:** You must restart the Connection Manager to start recording your customer login failure preferences. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

# load_transition_type

Use this utility to load custom deal and plan transition types into the BRM database. You define custom transition types in the *BRM_Home*/**sys/data/pricing/example/pin_transition_type** file.

For more information, see "Creating Custom Transition Types for Deals and Plans".

**Note:** You cannot load separate **/config/transition_type** objects for each brand. All brands use the same object.

**Caution:** When you run **load_transition_type**, it overwrites the existing **/config/transition_type** object. If you are updating your transition types, you cannot load new transition types only. You must load complete sets of transition types each time you run the utility.

**Important:** To connect to the BRM database, this utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## 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.

To verify that the transition types were loaded, you can display the **/config/transition_type** object by using Object Browser, or use the **robj** command with the **testnap** utility. See "Reading an Object and Writing its Contents to a File" in *BRM Developer's Guide*.

# pin_product_clear

Use this utility to purge the Audit Accounts Product table of data that is older than a specified time period. When you modify a product in Customer Center, BRM records the details of the product modifications in the AUDIT_ACCOUTNTS_PRODUCT_T table. BRM stores audit data when a customer purchases an item product and modifies or cancels an existing subscription product.

> **Note:** To connect to the BRM database, the **pin_product_clear** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## Location

*BRM_Home***/bin**

The **pin.conf** file for this utility is located in BRM_Home**/apps/pinapps/pin_rerate**. Run **pin_product_clear** from this directory.

## Syntax

```
pin_product_clear   -n days -d mm/dd/yyyy [-verbose] [test]
                    [-help]
```

## Parameters

**-n *days***
Deletes all the records from the Audit Accounts Product table that are older than the number of days specified by days.

**-d *mm/dd/yyyy***
Deletes all records from the Audit Account Product table that are dated prior to the midnight of the date you have specified.

**-verbose**
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 **-verbose** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
>
> **pin_product_clear** *other_parameters* **–v >** *filename***.log**

**-test**
Tests the utility, but does not affect the Audit Accounts Product table.

**-help**
Displays the syntax and parameters for this utility.

## Results

If the **pin_product_clear** utility doesn't notify you that it was successful, look in the utility log file (**pin_product_clear.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_state_change

Use this utility to perform bulk service state transitions 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. The utility finds services whose state expires on the current date and changes their state based on the applicable state transition.

> **Important:** If you use custom service life cycles, Oracle recommends that you run the utility daily.

In the **/config/lifecycle_states** object associated with a service, the default next state is the state in the **PIN_FLD_TRANSITIONS** array whose **PIN_FLD_DEFAULT_FLAG** field 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 service's state.

To get the applicable state transition, this utility calls the PCM_OP_CUST_GET_LIFECYCLE_STATES opcode.

To perform the state transition, this utility calls the PCM_OP_CUST_UPDATE_SERVICES opcode.

See "Managing Service Life Cycles" for more information on changing states in custom service life cycles.

> **Note:** To connect to the BRM database, this utility needs a configuration (**pin.conf**) file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

## 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 called the PCM_OP_CUST_UPDATE_SERVICES opcode to make a state change
- The number of state changes that were successful

For more information on why any failures occurred, see the log for PCM_OP_CUST_UPDATE_SERVICES.

# pin_unlock_service

Use this utility to unlock a locked CSR account. This utility also resets the CSR account password to a temporary one. The system administrator needs to provide the temporary password while unlocking the locked CSR account. For more information, see "Unlocking a Locked CSR Account" in *BRM System Administrator's Guide*.

**Important:**

- The **pin_unlock_service** utility needs a configuration (**pin.conf**) file in the same directory from which you run the utility. The **pin.conf** file required for running this utility is located in the *BRM_Home*/**apps**/**pin_unlock_service** directory. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.
- Make sure that the account in the **pin.conf** file is not locked.

## Location

*BRM_Home*/**bin**

## Syntax

```
pin_unlock_service
```

## Parameters

This utility accepts parameters through command prompts only. After each entry, press the **Enter** key to confirm your selection. For more information, see "Unlocking a Locked CSR Account" in *BRM System Administrator's Guide*.

## Results

This utility notifies you of all the results in the command console. Look in the **pin_unlock_service.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.