Oracle® Communications Billing and Revenue Management

Telco Integration

Release 7.5

E16721-12

August 2016

ORACLE®

Oracle Communications Billing and Revenue Management Telco Integration, Release 7.5

E16721-12

Contents

3 Installing Wireless Suite

Part II Prepaid AAA Overview

4 Understanding Prepaid AAA

5 Providing In-Session Notifications for Network Connectivity Applications

6 Managing Subscriber Preferences

7 How BRM Processes Prepaid AAA Requests

8 Using Lightweight Authorization

9 About Provisioning GSM Services

12 Installing GSM AAA Manager

13 Setting Up GSM Wireless Pricing

Part IV Services Framework Overview

14 Understanding the Services Framework

15 Installing Services Framework Manager

16 Installing Services Framework AAA Manager

17 About Customizing the Services Framework Manager Client

18 About Managing Prepaid Services and Extended Rating Attributes

19 About Provisioning Services

20 About Performing AAA for Prepaid Services

26 Installing GPRS Manager 3.0

27 About Managing and Provisioning GPRS Services

28 Installing GPRS AAA Manager

34 Installing and Configuring Number Manager and Number Administration Center

35 Number Manager Utilities

Part VIII Managing SIM Card Inventory

36 About Managing SIM Card Inventory

37 Installing and Configuring SIM Manager and SIM Administration Center

38 load_pin_sim_config

Part IX Managing Voucher Inventory

39 About Managing Voucher Inventory

44 Using the IP Address Manager APIs

Part XI Managing Dropped Calls and Continuation Calls

45 About Finding Dropped Calls and Continuation Calls

46 Configuring Your System for Dropped Calls and Continuation Calls

Preface

This guide describes how to integrate Oracle Communication Billing and Revenue Management (BRM) with your telco services.

Audience

This guide is intended for BRM system administrators and developers.

Downloading Oracle Communications Documentation

Product documentation is located on 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
E16721-01	November 2011	Initial release.

Version	Date	Description
E16721-02	May 2012	Documentation updates for BRM 7.5 Patch Set 1. ■ Added documentation for in-session notifications to the following: Understanding Prepaid AAA Providing In-Session Notifications for Network Connectivity Applications How BRM Processes Prepaid AAA Requests About Performing AAA for Prepaid Services ■ Added information about the VALIDATE_LIFECYCLE processing stage.
E16721-03	August 2012	Documentation updates for BRM 7.5 Patch Set 2. ■ Added "Services Framework AAA Utilities" chapter.
E16721-04	December 2012	Documentation updates for BRM 7.5 Patch Set 3. ■ Added documentation for offer profile threshold breach notifications to the following: Understanding Prepaid AAA Providing In-Session Notifications for Network Connectivity Applications How BRM Processes Prepaid AAA Requests Understanding the Services Framework About Performing AAA for Prepaid Services About Performing AAA for Prepaid GPRS Services
E16721-05	March 2013	Documentation updates for BRM 7.5 Patch Set 4. ■ Replaced multidatabase information with multischema information.
E16721-06	July 2013	Documentation updates for BRM 7.5 Patch Set 5. ■ Made minor formatting and text changes.
E16721-07	August 2013	On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5. Documentation added for HP-UX IA64.
E16721-08	October 2013	Documentation updates for BRM 7.5 Patch Set 6. ■ Added the "How BRM Refunds Charges for Prepaid Activity Events" section.
E16721-09	February 2014	Documentation updates for BRM 7.5 Patch Set 7. ■ Made minor formatting and text changes.

Version	Date	Description
E16721-10	May 2014	Documentation updates for BRM 7.5 Patch Set 8. ■ Added the "Configuring Service Status Change for Device-to-Service Associations" section.
E16721-11	December 2015	Documentation updates for BRM 7.5 Patch Set 14. ■ Updated the following sections: How BRM Authorizes Users to Access Prepaid Services How BRM Updates and Reauthorizes Prepaid Sessions How BRM Starts Prepaid Sessions How BRM Ends Prepaid Sessions
E16721-12	August 2016	Documentation updates for BRM 7.5 Patch Set 16. ■ Updated the following sections: Preparing GSM-Specific Data by Using Helper Opcodes

Part I

Overview of Integrating Wireless Services

Part I provides an overview of integrating wireless services in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

1

About Integrating Wireless Services

This chapter provides a conceptual overview of how to use your Oracle Communications Billing and Revenue Management (BRM) system to provide rating and customer management for wireless services.

Before reading this chapter, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

Important: Integrating your wireless services with BRM requires downloading and installing some or all of the following optional components:

- GSM Manager
- GSM AAA Manager
- Customer Center with GSM Manager Customer Center Extension
- Pipeline Rating Engine
- SIM Manager and SIM Administrator
- Number Manager and Number Administrator
- GPRS Manager

Note: Many of these components are bundled in Wireless Suite. See "Installing Wireless Suite".

About Using BRM for Wireless Services

The BRM wireless components manage customers and rate wireless usage events. Billing is handled by the standard BRM system.

You install and setup these components depending on the type of services you are offering.

GSM Services

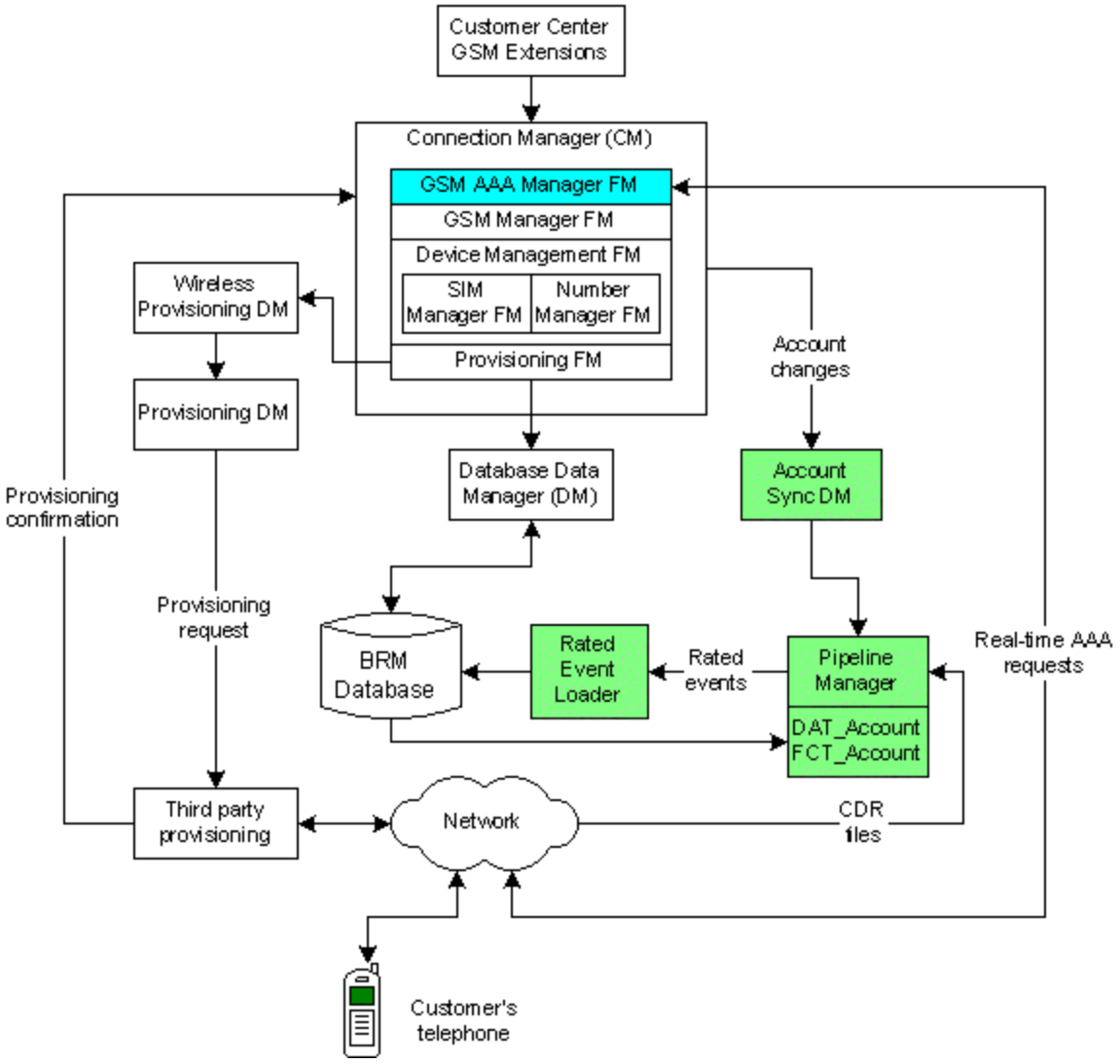
If you offer GSM services or other wireless services over the GSM network, install GSM Manager.

- To create and manage GSM accounts, use GSM Manager.

- To rate postpaid GSM services, configure batch rating by using Pipeline Manager and Rated Event (RE) Loader.
- To rate prepaid GSM services, configure real-time rating by using GSM AAA Manager.
- To manage GSM SIM cards and telephone numbers, use Number Manager and SIM Manager.

Figure 1–1 shows an overview of the BRM components in a GSM wireless implementation:

Figure 1–1 BRM Components in a GSM Wireless Implementation



GPRS Services

If you offer GPRS services, you have the following options:

- If your GPRS services are postpaid and large volumes of call detail records (CDRs) are provided from the network in files, use GSM Manager with Pipeline Manager and RE Loader to rate the CDRs.

Note: You can use GSM Manager to provide SIM card and telephone number support to GPRS users. However, you cannot use GSM Manager to support GPRS access point names (APNs). Therefore, if you are offering only GPRS data services, you can use GPRS Manager to manage GPRS services and accounts and rate usage. If GPRS events are provided in large volumes in files by the network, you must also install GSM Manager and rate the CDRs using Pipeline Manager and RE Loader.

- If your GPRS services are prepaid, or you otherwise require real-time rating for postpaid services, use GPRS Manager.
- If your GPRS services are postpaid and small volumes of call detail records (CDRs) are provided from the network in files, use UE Loader with GPRS Manager.

For more information, see "About Performing AAA for Prepaid GPRS Services".

About Managing SIM Cards

You use SIM Manager and SIM Administration Center to manage SIM cards. SIM Manager is a set of opcodes and storable classes required for creating and managing SIM card devices in the BRM database. SIM Administration Center is a GUI application that you use to create and process SIM card orders, and to change SIM card brand and network elements.

SIM cards are stored as devices in BRM.

Note: You can associate SIM cards only with GSM services.

You use Customer Center to assign and manage SIM cards in customer accounts. For more information, see the following topics:

- About Managing SIM Card Inventory
- Information about working with GSM accounts in the Customer Center Help.

About Managing Telephone Numbers

You use Number Manager and Number Administration Center to manage telephone numbers.

- Number Manager is a set of opcodes, utilities, configuration files, and storable classes required for managing GSM numbers.
- Number Administration Center is a GUI application used by operations personnel to manage telephone number inventory.

Note: You can associate telephone numbers only with GSM services.

You use Customer Center to assign and manage numbers in customer accounts. For more information, see the following topics:

- About Managing Telephone Numbers
- Information about working with GSM accounts in the Customer Center Help.

How Accounts Are Created and Managed

You use Customer Center to create GSM accounts.

To create accounts that use GSM services:

1. In Customer Center, choose a plan that includes GSM services, and assign a SIM card and a telephone number.

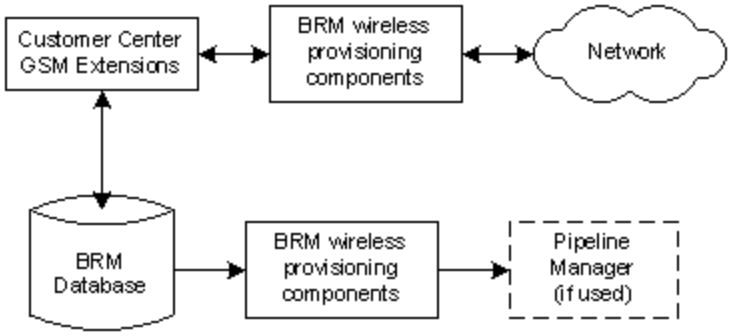
   Customer Center sends provisioning data to the provisioning system to provision the SIM card, number, and GSM service on the network.

2. (Optional) After the account is created, configure promotions such as friends and family discounts, if your plan includes them.

   If you installed Pipeline Manager to rate the CDRs, Account Synchronization Data Manager alerts Pipeline Manager that account information has changed. Pipeline Manager rating gets the account information from the BRM database.

   Figure 1–2 shows how GSM accounts are created and managed in BRM when using Pipeline Manager.

Figure 1–2 Using Pipeline Manager to Manage GSM Accounts



For more information, see the Customer Center Help.

How GSM Usage Events Are Rated

The GSM network sends usage events in call detail records (CDRs) to the BRM system. CDRs for *postpaid* GSM services are sent in files to Pipeline Manager. CDRs for *prepaid* GSM services are sent to GSM AAA Manager, which uses real-time rating.

Rating Postpaid GSM Events

When rating postpaid GSM events, BRM performs the following operations:

1. Pipeline Manager reads each CDR file and rates the CDRs using the following elements:

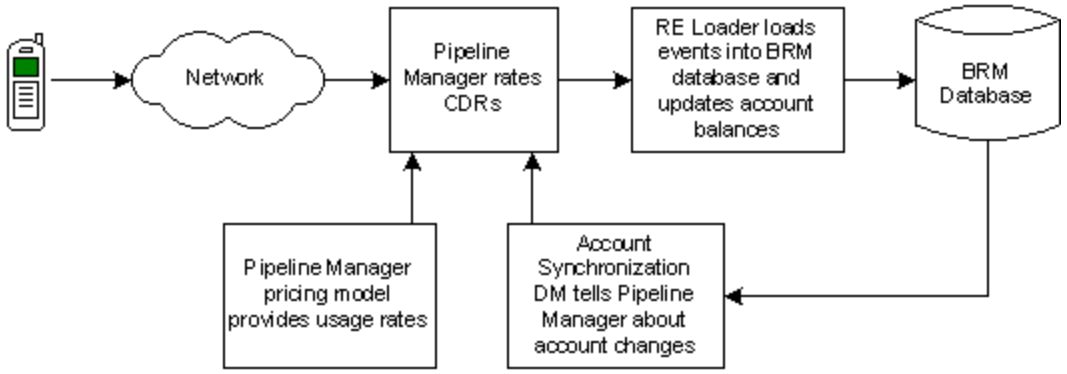
   - Pipeline rate plans.
   - Customer data obtained from the BRM database by using the Account Synchronization Data Manager.

Note: By default, pipeline rating identifies the customer's account by using the telephone number for the service being rated. You can customize pipeline rating to use any type of unique ID to identify an account.

2. Pipeline Manager creates an output file that includes a pre-rated event for the call.
3. RE Loader loads the event into the BRM database and updates the customer's account balance.

Figure 1–3 shows how a call is rated by Pipeline Manager:

Figure 1–3 How Pipeline Manager Rates Calls



Rating Prepaid GSM Events

External networks send AAA requests to GSM AAA Manager.

GSM AAA Manager then performs the following:

- Authenticates users by validating their device ID.
- Authorizes customers to access GSM services.
- Reauthorizes users to continue a GSM session.
- Records information about GSM sessions while they are in progress.
- When the session ends, rates the usage, debits the customer's prepaid balance, and records information about the session in the BRM database.

For more information, see "About Performing AAA for Prepaid GPRS Services".

2

Overview of BRM Wireless Services Installation

This chapter describes the tasks you need to perform to integrate Oracle Communications Billing and Revenue Management (BRM) wireless rating and customer management components.

You should read this chapter *before* installing and configuring the individual wireless components.

For basic information about integrating wireless services with BRM, see "About Integrating Wireless Services".

Overview of a BRM Wireless System

Before you install and configure a wireless system, you should understand how the components work together.

Creating and Managing Customer Accounts

You use the following components for creating and managing customer accounts:

- Use GSM Manager to add the storable classes and GSM FM opcodes that provide functionality for managing GSM services.
- Use SIM Manager and SIM Administrator to manage your SIM card inventory. The SIM FMs include opcodes that support SIM management. See "About Managing SIM Card Inventory".
- Use Number Manager and Number Administrator to manage your telephone number inventory. The Number FMs include opcodes that support number management. See "About Managing Telephone Numbers".
- Use a provisioning system to send GSM provisioning requests to the network.
- Use the GSM Manager Customer Center Extension with Customer Center to create and manage accounts that own GSM services. See information about working with GSM accounts in the Customer Center Help.

Setting Up the Price List

To set up your price list:

- Use Pricing Center to set up your GSM service rate plans.
- Use the sample GSM price list as an example when creating your GSM services price list.

Setting Up Rating

You use the following components to rate calls:

- If you are rating batch events with Pipeline Manager:
  - Use Pipeline Rating Engine to rate events, such as calls or data transfers. Pipeline Manager uses the **DAT_AccountBatch** and **FCT_Account** modules to retrieve data from the BRM database and apply it to events when rating.
  - For GSM services, use the Account Synchronization Data Manager (DM) to send updated account information to pipeline rating for call rating. See "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.
  - Use Rated Event (RE) Loader to import prerated events into BRM. See "Loading Prerated Events" in *BRM Configuring Pipeline Rating and Discounting*.
- If you are rating real-time events, use GSM AAA Manager. Rating is performed in real time by BRM rating opcodes.

For more information, see "About Integrating Wireless Services".

If you offer GPRS services, but not GSM services, you need to install only GPRS Manager for real-time rating.

Overview of Installation and Configuration Tasks

This section describes the tasks required to implement wireless services.

Important:

- Install, configure, and test each component before installing the next component.
- Be sure that the BRM system is running properly before installing optional components.
- Be sure that system and environment variable values do not change during installation of the components.

Note: You do not have to follow these steps in this exact order. However, some components need to be configured before others. See "Component Configuration Dependencies".

---

1. Install and configure BRM. This includes setting up standard BRM business policies, such as how to run billing and manage customers. If you use pipeline rating, set up your G/L IDs, rate plans, and resources before configuring pipeline rating.

   Important:

   - You must install BRM before installing GSM Manager 2.0.
   - Before installing and configuring any optional components, such as GSM Manager, run enough tests to ensure that your BRM core server components are installed and configured correctly.

See "Putting Together Your BRM System" in *BRM Installation Guide*.

2. Install SIM Manager. This includes:
   - Installing SIM Manager and SIM Administrator.
   - Completing the mandatory configuration tasks, for example, defining number device/service associations.

   See "Installing and Configuring SIM Manager and SIM Administration Center".
3. Install Number Manager. This includes:
   - Installing Number Manager and Number Administrator.
   - Completing the mandatory configuration tasks, for example, defining number device/service associations.

   See "Installing and Configuring Number Manager and Number Administration Center".
4. If you offer GSM services, install and configure GSM Manager and Wireless Provisioning Data Manager. See "Installing and Configuring GSM Manager and Provisioning Data Manager".
5. Install the GSM Customer Center Extensions. See "Configuring and Testing GSM Manager and Provisioning Data Manager".

   Important: Before you install GSM Manager Customer Center Extension, you must install GSM Manager and Customer Center.

6. If you offer GPRS services using GPRS Manager, install GPRS Manager and GPRS AAA Manager. See "Installing GPRS Manager 3.0" and "Installing GPRS AAA Manager".
7. If you are rating postpaid GSM services, see "Overview of Pipeline Manager Installation".
8. If you are rating prepaid GSM services, see "Installing GSM AAA Manager".
9. Install and configure the provisioning system.

Overview of Pipeline Manager Installation

1. Installing and configuring the Account Synchronization Data Manager (DM). The Account Synchronization DM allows pipeline rating to get data from the BRM database.
2. Running the **object_auditing.pl** script to turn on auditing for BRM objects that pipeline rating needs information about.

   See "Installing and Configuring the Account Synchronization DM" in *BRM Installation Guide*.

   Important: You cannot complete all the tasks required for sending account data to pipeline rating until you install pipeline rating. (For example, when you configure the Account Synchronization Data Manager, you need to specify the location of the Listener map file.)

3. Install and configure Pipeline Rating Engine. Before you configure rating, use the sample registry to test the system and make sure it has been installed correctly. See "Installing Pipeline Manager" in *BRM Installation Guide*.
4. Configure these pipeline components:
   - The pipeline "DAT_Listener," "DAT_AccountBatch," and "FCT_Account" modules
   - If you use Multidatabase Manager, the pipeline "FCT_AccountRouter" module

   See *BRM Configuring Pipeline Rating and Discounting*.
5. Install and configure Rated Event Loader to load events into the BRM database. See "Installing Rated Event Loader" in *BRM Configuring Pipeline Rating and Discounting*.

Summary of Installable Components

Table 2–1 summarizes the components you install for a typical wireless integration. This table assumes that you have installed the BRM server software and client applications.

Table 2–1 Installable Components

Installation Package	Description
BRM server software	Installs the standard BRM system software, including Connection Managers (CMs) and Data Managers (DMs).
Pipeline Rating Engine	Installs the Pipeline Rating Engine system software, modules, database, and utilities. The FCT_Account and DAT_AccountBatch modules are installed with the Pipeline Rating Engine.
GSM Manager	Installs the following: ■ GSM opcodes and storable classes. ■ Wireless Provisioning Data Manager. ■ Wireless provisioning opcodes.
GSM AAA Manager	Installs GSM AAA opcodes, storable classes, and utilities.
GSM Manager Customer Center Extension	Installs Customer Center components that support GSM services.
Account Synchronization Data Manager	Installs the Account Synchronization Data Manager (DM) and the **object_auditing.pl** script.
Rated Event Loader	Installs Rated Event Loader.
SIM Manager	Installs the SIM Manager server components.
SIM Administration Center	Installs SIM Administration Center.
Number Manager	Installs the Number Manager server components.
Number Administration Center	Installs Number Administration Center.
GPRS Manager	Installs GPRS Manager.

Component Configuration Dependencies

Some components cannot be configured without configuring another component first:

- Before you install any of the optional BRM GSM components, you must install the BRM server software.
- Before you install GSM Customer Center Extension, you must install GSM Manager and Customer Center.
- Before you install GSM AAA Manager, you must install GSM Manager.
- Before you create a SIM card inventory with pre-provisioned SIM cards, you need to set up provisioning.
- Before you create provisioning tags for extended rating attributes (ERAs), you need to configure ERAs in the pipeline.
- Before you create your GSM price list, you need to do the following:
  - Install GSM Manager.
  - Create and load provisioning tags. To use the sample GSM price list, you can load the sample **pin_gsm_provisioning_tags** file without editing it.
- Before you configure rating in the pipeline, you need to define the following in BRM:
  - rate plan names
  - G/L IDs
  - resources

Prerequisites for Running Components

- Before you can run the Account Synchronization DM, you need to install and configure the Pipeline Manager. See "Installing Pipeline Manager" in *BRM Installation Guide*.
- Before you run the **object_auditing** script, you must install your optional service components, such as GSM Manager and GPRS Manager.
- Before you run the pipeline with the DAT_AccountBatch and FCT_Account modules, you must run the **object_auditing** script to create the audit event tables that the DAT_AccountBatch and FCT_Account modules need. See "Turning on Object Auditing" in *BRM Installation Guide*.

Supported Operating Systems and Databases

All BRM wireless integration components run on HP-UX IA64, Linux, AIX, and Solaris operating systems. In addition, the GSM Manager Customer Center Extension, SIM Administrator, and Number Administrator run on Windows.

Rated Event (RE) Loader can load events only into an Oracle database.

Running Components on Different Machines

The pipeline and RE Loader should run on the same system. While it is possible to install RE Loader on a BRM system or the database system, you get better performance if you install it on the pipeline system. If the pipeline and RE Loader are on different systems, you need to map the pipeline output directories to a drive local to RE Loader.

For a test system, you should use three machines:

- BRM
- Pipeline Manager and Rated Event Loader

- Provisioning system

For a production system, you run BRM components on multiple machines. Use the standard BRM guidelines described in "Putting Together Your BRM System" in *BRM Installation Guide*.

End-to-End Testing

When you have installed all the components and have made sure that each is installed correctly and functioning, you can perform an end-to-end test of the entire wireless implementation. To do so, create an account and rate a usage event for that account.

Before creating an account, you need to do the following:

- Load the sample price list in BRM, and load the sample pricing data in the pipeline. See "Setting Up GSM Wireless Pricing". For information about installing Pipeline Manager, see "Installing Pipeline Manager" in *BRM Installation Guide*.
- Create sample SIM cards. See "Creating SIM Cards for Testing".
- Create sample telephone numbers. See "About Managing Telephone Numbers".

To create a GSM account, use Customer Center. Create an account using the sample GSM price list to make sure that the account owns a GSM service.

To rate a usage event, create a sample CDR file that uses the test account's phone number as the originating number. After loading the rated event, use Customer Center to make sure that the account balance is updated by the correct amount.

Important: In your sample CDR, the event dates must be later than the account creation dates.

By creating an account, you ensure the following:

- SIM cards and telephone numbers have been created.
- Services and ERA provisioning tags have been configured.
- The price list has been loaded.
- The provisioning system can receive and return provisioning requests.
- The Account Synchronization Data Manager and the Pipeline Listener can update the pipeline with a new account.

By rating a usage event, you ensure the following:

- The pipeline is configured correctly.
- The pipeline can get data from the BRM database.
- RE Loader is configured correctly.

3

Installing Wireless Suite

This chapter describes how to install and configure Oracle Communications Billing and Revenue Management (BRM) Wireless Suite.

About Wireless Suite

Wireless Suite is a set of optional features that you can install and configure in a single installation procedure.

Wireless Suite includes the following features:

- GSM Manager. See "About Integrating Wireless Services".
- GSM AAA Manager. See "About Performing AAA for Prepaid GSM Services".
- GPRS Manager. See "About Managing and Provisioning GPRS Services".
- GPRS AAA Manager. See "About Performing AAA for Prepaid GPRS Services".
- Number Manager. See "About Managing Telephone Numbers".
- RRF Manager. See "Reserving Resources for Concurrent Network Sessions" in *BRM Configuring and Collecting Payments*.
- Services Framework Manager. See "Understanding the Services Framework".
- Services Framework AAA Manager. See "About Performing AAA for Prepaid Services".
- SIM Manager. See "About Managing SIM Card Inventory".
- Voucher Manager. See "About Managing Voucher Inventory".

For information about configuring and using each feature, see the feature's documentation.

Note:

- You cannot choose which features to install. They are all installed.
- You can also purchase and install these features separately.

The installation procedure installs the software and loads sample configuration data into the database. For example, to configure Number Manager, the installation procedure runs the following utilities:

- The **load_pin_device_permit_map** utility, which loads the **pin_device_permit_map_num** file to create a **/config/device_permit_map** object.

- The **load_pin_device_state** utility, which loads the **pin_device_state_num** file to create a **/config/device_state** object.
- The **load_pin_num_config** utility, which loads the **pin_num_config** file to create a **/config/num object**.
- The **load_pin_network_elements** utility, which loads the **pin_network_elements** file to create a **/config/network_element** object.

After installing Wireless Suite, you have a system with sample data that you can use for testing. For your production environment, you need to configure each feature separately.

Note: Some of the load utilities used by the installation procedure overwrite existing configurations in the objects into which they load data. As a result, configurations loaded for one component of the suite may be overwritten during the installation of another component. For example, when the suite installs GSM Manager, the **load_pin_telco_provisioning** utility loads sample GSM data into the **/config/telco/provisioning** object. However, when the suite installs GPRS Manager, the **load_pin_telco_provisioning** utility overwrites the data in the **/config/telco/provisioning** object. To fix this situation, you must run the affected load utilities after installation to ensure that the correct configurations are loading into the relevant configuration objects.

System Requirements

Wireless Suite is available for the HP-UX IA64, Solaris, AIX, and Linux operating systems.

Software Requirements

Before installing Wireless Suite, you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM features. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM software.
- Oracle database software.

Important: To load the configuration data, you must install Wireless Suite on a system that has a Connection Manager installed on it.

Installing Wireless Suite

To install Wireless Suite:

1. Download the software.

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 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.

   Caution: 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 the temp_dir directory and enter this command:

   7.5.0_WirelessSuite_*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 is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or Wireless Suite is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the Wireless Suite 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.

7. If your event tables are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home*/**apps/partition_utils** directory:

   ```
   perl partition_utils.pl -o update
   ```

   For more information, see "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Setting Up Wireless Provisioning

To set up provisioning for the Wireless Suite Manager:

1. Enable wireless provisioning. See "Enabling Wireless Provisioning".
2. Set up event notification for provisioning, specify the details to add to service orders, and define the service order state transitions. See "Setting Up Services Framework for Provisioning".
3. To provision non-telco services, follow the instructions in "Setting Up Services Framework for Non-Telco Services".

Enabling Wireless Provisioning

By default, provisioning is disabled for Wireless Suite Manager.

To enable provisioning:

1. Open the Connection Manager (CM) **pin.conf** file in *BRM_Home*/**sys/cm**.
2. Change the value of the **provisioning_enabled** entry to 1.

   The default is **0** (disabled).
3. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Uninstalling Wireless Suite

To uninstall Wireless Suite, run *BRM_Home*/**uninstaller/WirelessSuite/uninstaller.bin**.

Part II

Prepaid AAA Overview

Part II provides an overview of integrating prepaid AAA services in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

4

Understanding Prepaid AAA

This chapter provides an overview of how your Oracle Communications Billing and Revenue Management (BRM) system performs authentication, authorization, and accounting (AAA) for prepaid services.

Before reading this chapter, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

How BRM and Your External Networks Support AAA

In prepaid environments, an external network and the BRM rating and billing system work together to establish voice and data connections in real time.

The external network collects information about the customer, connects the service, and collects data about the prepaid session while it is in progress.

BRM performs the following functions:

- Authenticates customers by verifying their identity. See "About Authenticating Prepaid Customers".
- Authorizes customers to use a specific service. See "About Authorizing Prepaid Usage".
- Reauthorizes customers for extended usage, if needed. See "About Reauthorizing Prepaid Usage".
- Cancels authorization for failed connections. See "About Canceling Authorizations for Prepaid Usage".
- Accounts for the results of prepaid sessions in progress. See "About Accounting for Prepaid Sessions".
- If your system is configured to receive in-session notifications from BRM (that is, when the **piggyback** business parameter is enabled), appends specific in-session notifications to the responses it provides for authorization and reauthorization requests sent by a supported network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".
- If your system is configured for policy-driven charging sessions, sends notifications to the network policy controller when the sum of the current balance and consumed reservation for a given service and resource ID matches the nearest threshold configured in the offer profile for that service and resource ID. (It supports such notifications for both in-session and out-of-session notifications). See "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.

When customers attempt to use a prepaid service, the external network collects information about the customer and sends authentication and authorization requests

to BRM. BRM processes the requests and returns the results immediately, so the network can connect the call.

After the service is connected, the external network begins collecting information about the customer's usage and sends the data to BRM, which records the data and rates the usage.

About Authenticating Prepaid Customers

BRM authenticates customers by comparing the ID that the customer provides with the ID stored in the BRM database. The type of ID BRM uses for verification depends on the service type:

- For telco services, such as GSM, the ID is typically a device ID, such as an MSID.
- For Internet Protocol (IP) services, such as email and Internet access, the ID is typically a login name and password.

BRM authenticates IDs in Password Authentication Protocol (PAP) mode or Challenge-Handshake Authentication Protocol (CHAP) mode.

BRM can perform additional verification checks before approving or denying an authentication request. For example, you can customize BRM to perform these authentication checks:

- **Credit limit checking**. BRM determines whether the customer's account balance currently exceeds the specified limit.
- **Service status checking**. BRM confirms that the requested service is currently active in the customer's account.
- **Duplicate session checking**. BRM checks for duplicate sessions.

For detailed information, see "How BRM Authenticates Prepaid Customers".

About Authorizing Prepaid Usage

BRM uses authorization to ensure that a customer is allowed to access a service, such as GSM telephony or SMS text messaging. Authorization is based on account information, such as the products a customer owns, the services a customer subscribes to, and the customer's current account balance.

BRM authorizes a customer to use a service for:

- A specified duration or volume. See "About the Authorized Duration or Volume".
- (Volume-based authorizations only) A specified validity period. See "About the Validity Period for Volume-Based Authorizations".

The authorization process includes these operations:

- Verifying that the customer has a subscription for the requested product or service.
- Determining whether the user has sufficient resources for the requested service or product. See "About Determining whether there Are Sufficient Resources".
- Calculating maximum authorizations if a request cannot be fully authorized. For policy-driven charging sessions, readjusting the quota accordingly. See "About Calculating Maximum Authorizations".
- Reserving a portion of the customer's resources for the prepaid session. See "About Reserving Resources for Prepaid Services".

- If your system is configured to receive in-session notifications from BRM (that is, when the **piggyback** business parameter is enabled), in-session notifications are appended to the responses returned the network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".

For detailed information, see "How BRM Authorizes Users to Access Prepaid Services".

> **Note:** If you use prepaid lightweight authorization, you can authorize customers who maintain a certain level of resources without requiring BRM to make calls to the rating and discounting engines. You can also *re*authorize customers without going through the rating process as long as they maintain the level of resources you specify. See "Using Lightweight Authorization".

About the Authorized Duration or Volume

Customers are authorized to use a service for a specified volume or duration. For example, customers can be authorized to download 100 bytes of data or make a 30-minute telephone call. The volume or duration that customers are initially authorized to use is specified in the authorization request from the network or based on configured default values.

- To pass in the authorization volume or duration to the BRM API, see "Authorizing Prepaid Services".
- To set default authorization values in BRM, see "Specifying Default Authorization and Reauthorization Values".

For prepaid sessions that are measured by more than one ratable usage metric (RUM), customers are authorized for all applicable resources. For example, a GPRS user might be authorized to use 10 free Anytime minutes and 25 megabytes for a session.

About the Validity Period for Volume-Based Authorizations

When rates are based on the time of day (TOD), BRM assigns a validity period to all volume-based authorizations. BRM sets the validity period to expire when the rates change. For example, if a customer accesses a service at 10:50 p.m. and the rates change at 11:00 p.m., the validity period is set to 9 minutes 59 seconds. At the end of the validity period, BRM forces a reauthorization at the new rate. This ensures that the authorization is valid for the current rate only.

For example, assume that a service has an authorization volume of 10 megabytes and the following simplified rate structure:

- From 6:00 a.m. to 6:59 a.m., $1 per megabyte
- From 7:00 a.m. to 8:59 a.m., $2 per megabyte

A customer that accesses the service at 6:55 a.m. will be authorized if the account balance is at least $10 (10 megabytes x $1 per megabyte) and the authorization will be valid for 4 minutes 59 seconds. At 6:59 p.m., BRM forces a reauthorization at a rate of $2 per megabyte and with a validity period of 119 minutes 59 seconds (the time of the next rate change).

Important: When BRM forces a reauthorization based on tariff changes, a spike in the network traffic occurs. You can reduce network spikes during a tariff change by delaying reauthorizations to distribute them more evenly. See "About Reducing Network Spikes during a Tariff Change".

About Determining whether there Are Sufficient Resources

To determine whether a customer has sufficient resources to use a service, BRM performs the following operations:

1. Estimates the cost of usage by rating the event with the authorization duration or volume, such as 30 minutes or 200 bytes, and optionally applying discounts. If multiple RUMs apply to the event, the estimate includes all of them.
2. Calculates the customer's available resources by retrieving current balances and subtracting any active reservations.
3. Compares the estimated cost against the customer's available resources.
   - If there are sufficient resources, BRM authorizes the usage.
   - If there aren't sufficient resources, BRM determines the maximum that can be authorized. See "About Calculating Maximum Authorizations".

For example, assume a customer with a prepaid balance of $50 wants to use a service that costs $1 per minute and is eligible for a 10% discount. If the authorization value is 20 minutes, BRM determines whether there are sufficient resources as follows:

1. Estimates the cost of usage by:
   - Rating the event: $1 per minute x 20 minutes = $20
   - Applying the 10% discount: $20 – $2 = $18
2. Calculates the customer's available resources: $50
3. Compares the estimated cost against the customer's available resources: $18 is less than $50.

Because the customer has sufficient resources, BRM authorizes the usage.

Determining Resource Sufficiency for Policy-driven Charging Sessions

For policy-driven charging sessions, BRM readjusts the quota in the following way:

1. Retrieves the balances for all the resources associated with the resource ID.
2. Retrieves all the associated offer profiles for the service POID and account POID.
3. Reduces the consumed quota by the consumed reserved amount in the Balances array.
4. Determines if the requested quota exceeds the amount available. If so, readjusts the quota to the amount that is available.

For example, the current balance on an account for a non-currency resource is 80 megabytes and the consumed reserved amount (across parallel sessions, iPhone, video, and computer) is 35 megabytes. When BRM receives a request to authorize a request for that resource and the requested amount exceeds 25 megabytes, BRM sets the allowable reservation quota for the session at 25 megabytes.

How BRM Handles Discountable Events and Multiple RUMs

The method BRM uses to determine sufficient resources depends on whether the event is discountable and whether it involves multiple RUMs:

- If an event is discountable or involves multiple RUMs, BRM first rates the event by using real-time rating opcodes. It then sends the information to a real-time discounting pipeline to apply discounts and charge sharing, calculate the customer's available resources, and compare the estimated cost against the customer's available resources.
- If an event is not discountable and does not involve multiple RUMs, BRM performs all steps: rating the event, calculating the customer's available resources, and comparing the estimated cost against the available resources: by using real-time rating.

About Calculating Maximum Authorizations

If a request cannot be fully authorized because of insufficient resources, BRM calculates the maximum amount the customer can use and authorizes that amount. For example, if a customer has a prepaid balance of $5 and wants to use a service that costs $1 per minute, the maximum amount for which the customer can be authorized is 5 minutes.

The effects of discounts, discount sharing, and charge sharing are included in the calculation of the maximum amount to authorize. For example, if the customer mentioned above is eligible for a 50% discount, the maximum usage that will be authorized is 10 minutes.

For more information about how BRM calculates maximum authorizations, see "Credit Limit Checks during Prepaid Authorization".

About Calculating Maximum Authorization for Policy-Driven Charging Sessions

For policy-driven-charging sessions, BRM readjusts the requested quota based on the current balance, used reservation across all parallel sessions, and the nearest threshold configured in the offer profile.

For example, the authorization request is for 30 MB. The current balance on the account for that resource is 80 MB and the consumed reserved amount (across parallel sessions, iPhone, video, computer,) amount to 35 MB. The next threshold in the offer profile is at 140 MB. The authorization process computes the allowable reservation quota at just 25 MB. BRM sets the provisioning for the session at 25 MB.

About Reserving Resources for Prepaid Services

When a prepaid session is authorized, BRM sets aside a portion of the customer's resources for the event. This prevents customers from using the resources for other services while the session is in progress. When the session ends, BRM rates the event based on the event's duration or volume and then returns any unused resources back to the customer's account balance.

For example, assume a customer has a prepaid balance of $50.

- When a prepaid session is authorized for $15, BRM reserves $15 for the session and leaves $35 that the customer can apply to other prepaid services.
- When the session ends and BRM determines the cost of the call to be only $5, BRM returns the remaining $10 to the customer's account balance. This updates the customer's account balance to $45.

For more information, see "Reserving Resources for Concurrent Network Sessions" in *BRM Configuring and Collecting Payments*.

About Reauthorizing Prepaid Usage

BRM authorizes prepaid usage for a specified duration or volume. However, customers may want to use their services beyond the authorized amount. When a prepaid event is in progress and the customer consumes most of the authorized amount, BRM can reauthorize the event for continued usage.

Reauthorization for prepaid services extends the following:

- The authorized duration or volume
- The validity period

The reauthorization process is similar to the authorization process, except BRM changes the duration or volume used during the verification process. The value used is the original authorization amount plus the specified extension amount. For example, if the original amount was 20 minutes and the specified extension amount is 10 minutes, BRM reauthorizes usage for 30 minutes.

The extension amount is specified in the reauthorization request from the network or is based on configured default values.

- To pass in the extension amount to the BRM API, see "Reauthorizing Prepaid Sessions".
- To set default reauthorization volumes and durations in BRM, see "Specifying Default Authorization and Reauthorization Values".

If the new value that is checked during reauthorization exceeds the customer's resources, BRM determines the maximum amount that can be reauthorized. BRM uses the same methodology for reauthorization that it uses for authorization. See "About Calculating Maximum Authorizations".

For volume-based services, BRM forces a reauthorization during a tariff change that causes a network spike. You can reduce network spikes during a tariff change by delaying reauthorizations to distribute them more evenly. See "Using Lightweight Authorization".

Note: If you use prepaid lightweight authorization, you can authorize customers who maintain a certain level of resources without requiring BRM to make calls to the rating and discounting engines. You can also reauthorize customers without going through the rating process as long as they maintain the level of resources you specify. See "Using Lightweight Authorization".

For policy-driven charging sessions, BRM sends a notification to the network policy controller when the sum of the current balance and used reservation reaches or crosses the nearest threshold configured in the offer profile for a given service and resource ID. This notification contains information about the policy label that is applicable, the current balance, and the difference in the amount between the current balance to the next label. If necessary, the network policy controller works with BRM to set up a new provisioning policy based on the current state of the subscriber's profile and usage amount. See "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.

About Canceling Authorizations for Prepaid Usage

After a session is authorized, the external network is sometimes unable to connect the service. This can occur because:

- The call's destination was unavailable.
- The validity period expired before the service was connected.
- The customer terminated the session before the service was connected.

In this situation, BRM can cancel the authorization and return any reserved resources back to the customer's account balance.

About Accounting for Prepaid Sessions

After a prepaid session is authorized, the external network connects the call and begins collecting information about any usage, such as the duration of the session, the time of day the session occurred, and the amount of data sent or received. The external network sends this information to BRM, which records it in the BRM database or in In-Memory Database (IMDB) Cache.

When a session ends, the BRM accounting process uses this information to determine how much to charge customers for the services they used.

BRM performs the following operations during the accounting process:

- Starts recording information about a prepaid session.
- Updates information about an existing prepaid session.
- Stops the prepaid session when the following occurs:
  - The session ends successfully.
  - The external network encounters a severe problem, shuts down abnormally, or restarts.
- Sends information about the completed session to the real-time rating opcodes, which perform the following:
  - Rate the customer's usage.
  - Update the customer's prepaid account balance.
  - Close the associated reservation and return any unused resources back to the customer's prepaid account balance.
  - Store information about the session in the BRM database.
- For policy-driven charging sessions:
  - Receives the message from the network connectivity application containing information on the service type, device ID, details about the call, and the consumed quota. It responds to the network connectivity application.
  - Sends a notification if there was a threshold breach.

  See "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating* for information.

For more information, see "How BRM Manages Prepaid Sessions".

Setting Up BRM to Process AAA Requests

To set up your system to process prepaid AAA requests for telco services, see "Setting Up a System Based on Event-Type Rating" in *BRM Installation Guide*.

5

Providing In-Session Notifications for Network Connectivity Applications

This chapter describes how you can provide improved notifications in real time when performing authentication, authorization, and accounting (AAA) for prepaid services using Oracle Communications Billing and Revenue Management (BRM) in conjunction with a network connectivity application.

About AAA Responses Containing In-Session Notifications

By default, the AAA response to an authorization or reauthorization request for a service contains the result (pass or fail) of the request and its reason and other details, such as current balances and reservations. You also have the option of providing, in real time, additional time-critical information about the request in these AAA responses. This additional information enables you to notify the subscriber, who can then take appropriate action (in real time).

For example, a subscriber makes a request to download a video. The network receiving the video download request forwards the request to the network connectivity application, which then sends the request to BRM. BRM processes the request and adds information on the subscriber's streaming usage threshold in its response to the network application. The network informs the subscriber that the streaming usage threshold is reaching its limit. The subscriber takes the necessary action in real time to address the situation.

The notifications that BRM appends to the responses for authorization and reauthorization requests from the network are called in-session notifications.

Such online charging functionality is made possible when you use BRM in conjunction with a network connectivity application (such as the Online Mediation Controller component of Oracle Communications Service Broker).

Note: The in-session notifications described in this chapter are different from the offer profile threshold breach notifications BRM sends to the network policy controller.

BRM supports offer profile threshold breach notifications when offer profiles are included in price lists for policy-driven charging for services. They are sent when a subscriber's usage of a resource crosses a threshold set in the offer profile for the service and purchased plan.

For more information, see "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.

About the Flow of Information Between BRM and the Network Connectivity Application

Information between BRM and the network connectivity application flows in the following way:

1. When a subscriber attempts to access a service, the network sends an authorization (or reauthorization) request to the network connectivity application.
2. The network connectivity application analyzes the request from the network, translates the information as necessary for BRM, and calls an appropriate AAA opcode to process the request.
3. BRM processes the request by using the details in the service request and the information in the subscriber's account (such as the current balances, credit thresholds, subscription information, and so on). BRM uses the subscriber profile information in the subscriber profile repository and appends in-session notifications about the request to the AAA response.
4. BRM sends the response back to the network connectivity application.
5. The network connectivity application uses the additional appended information in the response to provide appropriate pre-call and mid-call announcements to the the subscriber.
6. The subscriber takes the appropriate action based on the announcement he receives from the network.

About In-Session Notifications in BRM

In-session notifications are optional features in the AAA responses provided by BRM to network connectivity applications. For more information, see "Enabling In-Session Notifications".

About the Conditions that Trigger In-Session Notifications

The following conditions in processing AAA requests trigger in-session notifications:

- When the current balance together with the reserved amount reaches the credit threshold configured for the resource in the purchased plan (in Customer Center)
- When the current balance in a customer's account and the reserved amount of the configured resource together exceed the streaming usage amounts maintained as a non-currency usage counter for that account
- When the subscription for a requested service is about to expire.
- If a tariff change is impending for the resource in the AAA request, BRM sets up a tariff change indication in the response to prevent a spike in the number of reauthorization requests.

> **Note:** When a call request is denied, the AAA response for that call request contains the details associated with the call denial only. BRM does not append any in-session notification to such a response. See "How BRM Provides Call Denial Notifications".

About the Maximum Scaled Delay Time for Tariff Changes

The maximum scaled delay time for a resource is the maximum delay that can be returned to the caller for that resource. The higher the value of the maximum scaled delay time, the longer the scaled delay and, thus, the later the reauthorization. If you

have a large customer base, you may need to increase the maximum scaled delay time to spread out the reauthorizations over a longer period of time and thereby reduce network spikes more effectively.

BRM stores the maximum scaled delay time value in the **MaxScaledDelayTime** field for each resource ID in the **/config/aaa** object.

About the Opcode Used to Set Up In-Session Notifications

The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode sets up in-session notifications for one or more of the following, using the details in the AAA request and the current data in the subscriber's account:

- If there was a credit threshold breach for a requested resource, the policy opcode places the balance details for that resource in the notification.

  See "How BRM Provides Credit Threshold Breach Notifications".

- The opcode places the timestamp for when the service will expire as an in-session notification and enters the current time in the subscriber's preference.

  See "How BRM Provides Subscription Expiration Notifications".

- If the resource is being monitored for streaming usage and the streaming usage threshold was breached, the policy opcode enters the breach information as an in-session notification.

  See "How BRM Provides Streaming Usage Threshold Summaries".

- If there is an impending tariff change, the policy opcode calculates the scaled delay time for the resource and uses it to calculate when the tariff change is to occur.

  See "How BRM Provides Tariff Change Indication in the AAA Responses".

The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode returns the information it received from the calling opcode along with the created in-session notifications and the subscriber's preferences for announcements.

About Service-Related Information in the In-Session Notifications

In-session notifications appended to AAA responses contain information on one or more of the following:

- Credit threshold breach notifications contain information on current balances, credit thresholds, and so on.
- Streaming usage and credit summaries contain information on the current balance for the resource that crosses the subscriber's usage limit.
- Subscription expiration notifications contain the service expiration time from the **/service** object.
- Notifications of impending tariff changes for a service specify when the validity period ends for that service (the time when the tariff change is to take place).

See "How BRM Provides In-Session Notifications" for the AAA opcodes that provide this information. For the data contained in the in-session notifications, refer to the output flist of these AAA opcodes in *Opcode Flist Reference* in the BRM documentation.

About Subscriber's Preferences Data in In-Session Notifications

The following information from a subscriber's **/profile/subscriber_preferences** object are sent in the in-session notifications appended to the responses for the subscriber's requests:

- Preferred language
- Preferred channel
- Preferred time

In addition to this, the **/profile/subscriber_preferences** object contains information used to set up in-session notifications. See "Managing Subscriber Preferences" for more information.

Configuring Your Environment to Provide In-Session Notifications

To configure your environment to provide in-session notifications:

1. Enable in-session notifications in BRM.

   See "Enabling In-Session Notifications".

2. Provide the maximum scaled delay time allowed for tariff changes.

   See "About Configuring the Maximum Scaled Delay Time for a Resource".

3. Set up subscriber preferences data.

   For information on setting up subscriber preferences data using Customer Center, see "Maintaining Subscriber Preferences with Customer Center".

   For information on setting up subscriber preferences data using APIs, see "Maintaining Subscriber's Preferences Data with Custom Client Applications".

4. To provide subscription expiration notifications to your subscribers, enable custom service life cycles in BRM.

   See "Enabling BRM to Use Custom Service Life Cycles" in *BRM Managing Customers*.

For information on how to customize the handling of in-session notifications from BRM, see "Customizing the Handling of In-Session Notifications from BRM".

Enabling In-Session Notifications

To enable in-session notifications, enable the **Piggyback** business parameter in the following way:

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 **AAA** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsAAA bus_params_AAA.xml
   ```

   This command creates the XML file named **bus_params_AAA.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_AAA.xml.out** file.

4. Search for the following line.

```
<Piggyback>disabled</Piggyback>
```

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

   Caution: BRM uses the XML in this file to overwrite the existing **AAA** 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 configurations.

6. Save this file as **bus_params_AAA.xml**.
7. Go to the *BRM_Home*/**sys/data/config** directory.
8. Load this change into the appropriate **/config/business_params** object by running the following command:

   ```
   pin_bus_params PathToWorkingDirectory/bus_params_AAA.xml
   ```

   where *PathToWorkingDirectory* is the directory in which **bus_params_AAA.xml** resides.

   Note: To run this command from a different directory, see the description for **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 instruction 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 Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

About Configuring the Maximum Scaled Delay Time for a Resource

To configure the maximum scaled delay time for a specific resource, edit the appropriate AAA parameters XML file from among the following files located in the *BRM_Home*/**sys/data/config** directory.

- pin_telco_aaa_params.xml
- pin_telco_gprs_aaa_params.xml
- pin_telco_gsm_aaa_params.xml
- pin_telco_gsm_data_aaa_params.xml
- pin_telco_gsm_fax_aaa_params.xml
- pin_telco_gsm_sms_aaa_params.xml
- pin_telco_gsm_telephony_aaa_params.xml

You then load each updated file into the BRM database's **/config/aaa** object by running the **load_pin_telco_aaa_params** utility.

See "Configuring Services Framework AAA Parameters XML Files" for a description of how to edit and load the AAA parameters XML file into the BRM database.

How BRM Provides In-Session Notifications

BRM provides in-session notifications in the responses to AAA requests received during a prepaid session and when called upon to end the prepaid session.

During a prepaid session, the network connectivity application calls one of the following opcodes depending on whether the request requires an authorization or reauthorization:

- PCM_OP_TCF_AAA_AUTHORIZE
- PCM_OP_TCF_AAA_REAUTHORIZE
- PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE

When the network ends the prepaid session, the network connectivity application calls PCM_OP_TCF_AAA_STOP_ACCOUNTING, the main opcode for ending prepaid sessions.

Each of these opcodes calls the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode at the end of its process. Each opcode provides the service or the account information from the processing of the request in its input to the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode. The policy opcode returns the input values with the appropriate in-session notifications appended to them. The calling opcode receives the output from the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode and sends that information to the network connectivity application.

See "How BRM Processes Prepaid AAA Requests" for more information on the individual processes.

How BRM Provides Credit Threshold Breach Notifications

BRM provides credit threshold breach information in the in-session notifications if you have configured the necessary credit threshold information in the plans you offer your customers. For more information on credit thresholds, see "About applying credit limits to resources" in *BRM Setting Up Pricing and Rating*.

BRM calculates the charges for the required resource by using the PCM_OP_RESERVE_CREATE and the PCM_OP_RESERVE_EXTEND opcodes. After the charges are calculated, BRM compares the sum of the customer's balance and reserved amount for the required resource against the customer's credit threshold value. If this amount crosses the credit threshold value, BRM generates a notification.

For more information, see "Managing Customer Billing Information" in *BRM Managing Customers*.

How BRM Avoids Repetitions of Threshold Breach Notifications for a Session

To avoid multiple threshold breach notifications for the same resource, BRM maintains a list of the last credit thresholds breached for the current session in an **/active_session** object for each session. The PCM_OP_ACT_AUTHORIZE and PCM_OP_ACT_REAUTHORIZE opcodes record the credit breach information on the last credit thresholds breached for the current session in the **/active_session** object. See "How BRM Processes Prepaid AAA Requests" for more information.

How BRM Provides Subscription Expiration Notifications

BRM provides subscription expiration notifications if you have enabled custom service life cycles. See "Enabling BRM to Use Custom Service Life Cycles" in *BRM Managing Customers*.

The PCM_OP_TCF_AAA_POL_POST_PROCESS policy pocked sets up a subscription expiration notification in the following way:

1. It retrieves the following information:
   - The date and time when the current custom service state expires (PIN_FLD_SERVICE_STATE_EXPIRATION_T from the **/service** object).
   - The subscriber's preferences from the **/profile/subscriber_preferences** object:
     - The number of days for which the expiration notification must be sent (**NotificationExpiry**)
     - The interval between notifications in days (**NotificationInterval**)
     - The date and time when the last notification was sent for this service (**SentNotificationTime**)

   The opcode uses the date (without the time) from the timestamp value in all the associated time-related objects.
2. The policy opcode computes the date when the subscription expiration notification must start by using PIN_FLD_SERVICE_STATE_EXPIRATION_T in the corresponding **/service** object and **NotificationExpiry**.

   For example, if the current service state for a requested service expires on March 16 and the expiration notification must be sent for 8 days prior to the service expiration, the start date for the notification is March 8.
3. The policy opcode determines whether the current date belongs to the set of dates requiring a notification by applying the **NotificationInterval** value to the start date.

   Continuing with the example, if the start date for the notification is March 8 and the customer requires a 3-day interval between notifications, then, at this time, the dates when notifications must be sent are March 8, March 11, and March 14.

   The policy opcode does one of the following:
   - If the current date requires a notification (in our example, March 8, March 11, or March 14) and the **SentNotificationTime** field of the **/profile/subscriber_preferences** object is empty:
     - The policy opcode stores the current time in the **SentNotificationTime** field of the **/profile/subscriber_preferences** object. This entry now acts as the time the last notification was sent for this subscription.
     - The policy opcode places a description (for example, "Service Expiration") in the PIN_FLD_DESCR field and the current custom service state expiry time (from the **/service** object) in the PIN_FLD_EXPIRATION_T field of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array.
   - If the current date is not one that requires a notification (in our example, it is not March 8, March 11, or March 14), the policy opcode does not set up any notification.

   The AAA opcode that receives the output from the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode forwards this information to the network connectivity application for setting up and delivering the actual announcement to the subscriber.

How BRM Provides Streaming Usage Threshold Summaries

BRM provides streaming usage threshold summaries if you have configured the necessary non-currency counters in the rate plans you offer your customers. For more information on setting up non-currency counters in rate plans, see the discussion on setting up real-time rate plans in Pricing Center Help.

When it is called to process an authorization or reauthorization request, the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode calculates the sum of the current balance for the non-currency usage amount and the amount to be reserved for this session. The opcode compares this computed value against the usage limit stored for this resource in the **StreamingThreshold** entry located in the subscriber's **/profile/subscriber_preferences** object.

If the computed value exceeds the usage limit, the policy opcode enters "Streaming Threshold reached" in the PIN_FLD_DESCR field and the current balance in the PIN_FLD_CURRENT_BAL field of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array.

How BRM Provides Call Denial Notifications

A call request can be denied by any of the following:

- The opcode authorizing or reauthorizing the request: When the denial originates from the opcode authorizing or reauthorizing the request, the PIN_FLD_ERR_BUF or PIN_FLD_RESULT field in the output flist from the opcode contains the reason for the error.
- Reservation framework: When resources cannot be reserved for the requested service, the PIN_FLD_REASON field of the output flist from the opcode authorizing or reauthorizing the request contains the reason for the rejection.
- The rating process: When there is a failure in rating the event, the PIN_FLD_RATING_STATUS field contains the rating status entry. To interpret the rating status entry, see the *BRM_Home***/include/pin_rate.h** file.

The opcode processing such a request from the network connectivity application sends the appropriate error code for the reason the call was denied as its output. In-session notifications are not included with any call denial response.

Call Denial Error Codes

Table 5–1 lists the possible call denial error codes and their values. See *BRM System Administrator's Guide* for information on troubleshooting.

Table 5–1 Error Values Returned for Call Denials

Field ID Containing Error Code	Error Codes
PIN_FLD_REASON	Error associated with reserving the resource: ■ 0 (Authorization failure) ■ 1 (Authorization success) ■ 2 (Duplicate reservation found) ■ 3 (Insufficient funds; partial reservation) ■ 4 (No funds available) ■ 5 (Insufficient Rated quantity) ■ 6 (Invalid requested quantity)

Table 5–1 (Cont.) Error Values Returned for Call Denials

Field ID Containing Error Code	Error Codes
PIN_FLD_RATING_STATUS	Status of quantity-based requests based on the rating process: ■ 0 (Rating Successful) ■ 1 (Zero quantity) ■ 10 (No Scale to rate) ■ 11 (No Candidate RUM) ■ 12 (No Initial Products) ■ 13 (Calc Max in Multi-RUM) ■ 14 (No Matching RUM) ■ 15 (No qualified products) ■ 16 (No RUM) ■ 17 (Status Mismatch) ■ 18 (Product not in DB) ■ 19 (No product in audit) ■ 20 (No rate plan) ■ 21 (No matching selector data) ■ 22 (No rating currency) ■ 23 (No valid rate span) ■ 24 (No valid rate) ■ 25 (No matching impact) ■ 26 (Credit limit exceeded)
PIN_FLD_RESULT	Session or service related errors found: ■ 0 (Authorization failure ■ 0 (Authorization success
PIN_FLD_AUTHORIZATION_ID	PIN_ERR_DUPLICATE (An active session with the authorization ID already exists in database)
PIN_FLD_MSID	PIN_ERR_BAD_ARG (Service not found)
EBUF	PIN_ERR_CREDIT_LIMIT_EXCEEDED (Credit limit exceeded)

How BRM Provides Tariff Change Indication in the AAA Responses

BRM calculates and provides the tariff change time for a request if you have specified the maximum scaled delay time allowed before the tariff changes for the resource by configuring the **MaxScaledDelayTime** entry for the resource Id in the **config/aaa** object. See "About Configuring the Maximum Scaled Delay Time for a Resource".

BRM provides the maximum scaled delay time allowed for tariff changes for the resource in the PIN_FLD_VALID_TO entry of the notification. For example, the entry Thu Feb 16 05:12:07 2012 in the notification of a tariff change means that the old rates for the resource apply up to that time. After Thu Feb 16 05:12:07 2012, new rates apply, and the service request requires a reauthorization.

Network connectivity applications can use the tariff change time in the in-session notifications to prevent different sessions from returning for reauthorization at the same time.

How BRM Calculates the Scaled Delay Time for a Service

If a tariff change for a requested service is impending, BRM calculates the scaled delay time as a factor of the maximum scaled delay time (**MaxScaledDelayTime**) configured for that resource ID in the **/config/aaa** object.

Note: If you have not configured **MaxScaledDelayTime** for that resource ID, BRM does not calculate or create a tariff change indication in the AAA response.

The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode calculates the scaled delay time value as shown in Figure 5–1:

Figure 5–1 Scaled Delay Time Calculation

The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode returns the computed scaled delay time in the PIN_FLD_VALID_TO field of the PIN_FLD_PIGGYBACK_NOTIFICATIONS array.

For more information on tariff change, see "Using Lightweight Authorization".

How BRM Provides Subscriber Preferences in Notifications

BRM sets up individual in-session notifications for the subscriber's preference for language, channel of communication and the preferred time for the announcements. It retrieves each of these values from the **/profile/subscriber_preferences** object associated with the account.

How the PCM_OP_TCF_AAA_POL_POST_PROCESS Policy Opcode Works

The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode prepares the in-session notifications in the following way:

1. It retrieves the subscriber's current preference information by calling the PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES opcode. See "Maintaining Subscriber's Preferences Data with Custom Client Applications".
2. If you have enabled custom life cycles in BRM, it checks on the subscription status. If the subscriber's subscription is about to expire, a check is made to determine if a notification was already sent.
   - If a notification was not sent during this session, the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode adds the subscription expiration notification to the output intended for the network connectivity application.
   - The policy opcode updates the subscriber's preferences with the time the notification was sent.

   See "Managing Custom Service Life Cycles" in *BRM Managing Customers*.

3. It checks whether the request is breaching the set threshold configured for the resource. If there is a threshold breach, the policy opcode appends an in-session notification for the breach. See "How BRM Provides Credit Threshold Breach Notifications".
4. If a streaming threshold usage notification was configured for the service or the account:
   a. The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode retrieves the current balance and the reservation amount.
   b. It checks whether there are any breaches for the resource.
   c. It appends the streaming threshold counter in the in-session notification.

   See "How BRM Provides Streaming Usage Threshold Summaries".
5. The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode calculates the scaled delay time and appends the tariff change for the resource in the PIN_FLD_VALID_TO field of the in-session notification.

   See "How BRM Provides Tariff Change Indication in the AAA Responses".
6. The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode sets up the subscriber's preferences as in-session notifications to the response.

   See "How BRM Provides Subscriber Preferences in Notifications".
7. The PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode sends the updated in-session information to the calling opcode. For more information on this policy opcode, see *Opcode Flist Reference* in the BRM documentation.

Customizing the Handling of In-Session Notifications from BRM

You can customize how you handle in-session notifications from BRM in the following ways:

- Customize the configuration of the subscriber preferences data. Customer Center reads the configurations in the **/config/subscriber_preferences_map** object to dynamically list the preferences that a subscriber can configure in the Preferences Tab. See "Customizing Subscriber Preferences" for more information.
- Customize the information provided to the network connectivity application about the customer's service or account.

  See "Customizing AAA Processes by Extending Policies".
- Access and act upon credit threshold breaches using the **testnap** utility or external applications. See "About Setting Up Custom Applications to Receive Credit Threshold Breach Information".

Customizing AAA Processes by Extending Policies

A customer can be allocated a custom bandwidth, and that custom data has been stored in the **/profile/subscriber_preferences** object. You can read in this information and add it to the AAA response that the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode sends to the network connectivity application.

To customize the processing of the AAA requests by extending the policies associated with AAA requests:

1. Retrieve the subscriber's preference data by calling the PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES opcode.

2. Perform your operations.
3. Update the data stored for the user by calling the PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES opcode.
4. Return the values in the output flist of the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode.

See *Opcode Flist Reference* in the BRM documentation for more information on the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode.

About Setting Up Custom Applications to Receive Credit Threshold Breach Information

You can set up the **testnap** utility or custom applications that you create to receive credit threshold breach information from the following AAA opcodes:

- PCM_OP_ACT_AUTHORIZE
- PCM_OP_ACT_REAUTHORIZE
- PCM_OP_RESERVE_CREATE
- PCM_OP_RESERVE_EXTEND

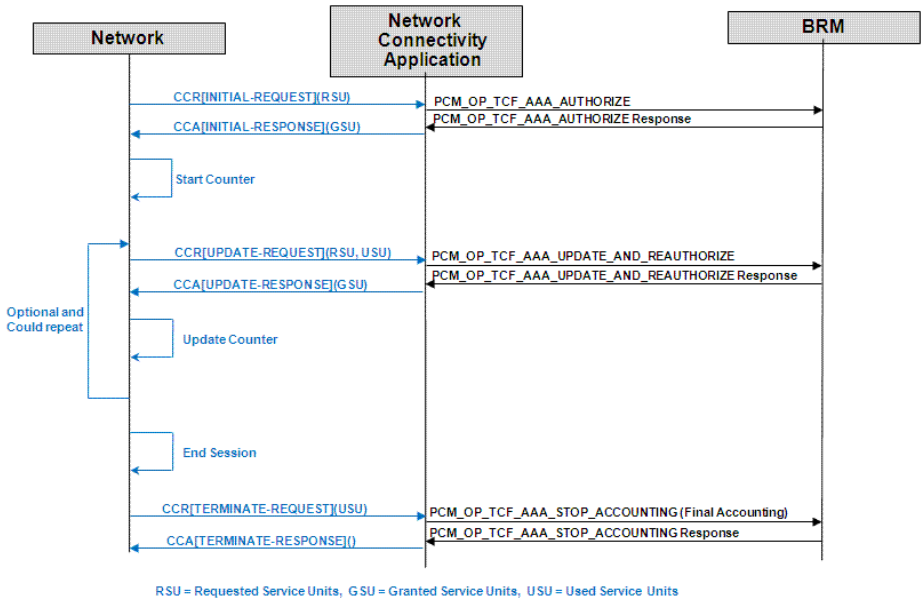
To receive credit threshold information from any of the above opcodes, set the PIN_FLD_PIGGYBACK_FLAG field to **enabled** in the input flist when you call the specific opcode.

See *Opcode Flist Reference* in the BRM documentation, for more information on these opcodes.

How Network Connectivity Applications Orchestrate Prepaid Sessions

Network connectivity applications use the information in the authorization or reauthorization request to select the logic appropriate to orchestrate a prepaid session and call the appropriate AAA opcode in BRM. Figure 5–2 shows how the flow works for a prepaid session:

Figure 5–2 Session-Based Request Flow



Before the start of the session, the network connectivity application calls PCM_OP_TCF_AAA_AUTHORIZE with the necessary call details and subscriber information required by BRM to authorize the session. BRM does the following:

- It checks the subscriber's profile and the account balances and responds by authorizing the session (or denying the request).
- It appends notifications (associated with the service, credit thresholds, any tariff indications, and so on) to the response.
- It returns the response.

Based on this response, the network connectivity application may set up appropriate pre-call announcements to be delivered to the subscriber.

For the duration of the session, there may be updated requests from the network. These updated requests trigger the network connectivity application to call PCM_OP_TCF_AAA_UPDATE _AND_REAUTHORIZE with the necessary call details and subscriber information required by BRM. BRM does the following:

- It checks the subscriber's profile and the account balances and responds by authorizing the session (or denying the request).
- It appends the appropriate mid-call notifications in its response.
- It returns the response.

The network connectivity application converts these notifications into appropriate mid-call announcements to be delivered to the subscriber.

When the session ends, the network sends a request to the network connectivity application to terminate the session. The network connectivity application calls the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode to end the prepaid session. BRM records the usage event in the database and appends any in-session notifications in the response it sends to the network connectivity application.

6

Managing Subscriber Preferences

This chapter describes how you can manage the subscriber preferences data in the subscriber profile repository offered by Oracle Communications Billing and Revenue Management (BRM).

About Subscriber Preferences

BRM allows you to manage how each subscriber prefers to receive notifications from the network. For example, you can specify that a subscriber wants to receive notifications in French via SMS text messages.

By default, BRM allows you to manage the following subscriber preferences:

- Preferred channel of communication: IVR, SMS, e-mail, and so on
- Preferred language of communication: English, French, and so on
- Number of days prior to which customer wishes to receive the notification
- Interval between two successive notifications
- Threshold value for streaming usage allowed for the subscriber
- Resource ID for tracking threshold breaches (in case of streaming usage)
- Timestamp of the last notification sent to the subscriber

BRM stores information about each subscriber's preferences in a subscriber profile repository. BRM stores the types of preferences that you track and their default values in the **/config/subscriber_preferences** object. BRM stores each subscriber's preferences at the account level and the service level in individual **/profile/subscriber_preferences** objects.

For more information on the **/config/subscriber_preferences** and **/profile/subscriber_preferences** objects, see *BRM Storable Class Reference*.

Maintaining Subscriber Preferences with Customer Center

When in-session notifications are enabled, you can configure and maintain subscriber preferences by using Customer Center. During the account creation and modification process, you specify the subscriber preferences in the Customer Center **Subscriber Preferences** page. See the Customer Center Help for more information.

Customer Center uses the configurations in the **/config/subscriber_preferences_map** object to dynamically list the preferences that a subscriber can configure. You can customize the information as necessary. See "Customizing Subscriber Preferences".

About Regulating Permissions to Update Subscriber Preferences

By default, all customer service representatives (CSRs) can access and update subscriber preferences. You can restrict a CSR's permissions to view and update a subscriber's preferences for services and accounts. For more information, see Permissioning Center Help.

Maintaining Subscriber's Preferences Data with Custom Client Applications

You can customize your external application to manage subscriber preferences during the account creation process by using the following opcodes:

- PCM_OP_CUST_SET_SUBSCRIBER_PREFERENCES

  This opcode creates, modifies, and deletes the **/profile/subscriber_preferences** object, which contains a subscriber's preference data. You can modify a specific preference or all of the subscriber's preferences.

- PCM_OP_CUST_GET_SUBSCRIBER_PREFERENCES

  This opcode retrieves the subscriber's preferences from the **/profile/subscriber_preferences** object.

  For more information, see *BRM Opcode Flist Reference*.

Customizing Subscriber Preferences

To customize the subscriber profile data configuration by using **config_subscriber_preferences_map.xml** file:

1. Open the *BRM_Home***/sys/data/config/config_subscriber_preferences_map.xml** file.
2. Edit the file which includes examples and instructions. Table 6–1 describes the parameters in this file:

Table 6–1 Elements Used to Store a Subscriber Preference

Element	Description
Name	Name of the preference
ID	The ID associated with the preference
Type	The type of value that the preference can be assigned, from one of the following types: ■ **1**: STR (alphanumeric) ■ **2**: INT (integer) ■ **3**: ENUM (indicating that the preference is one of an ordered list of possible values. An array of values must be provided for this selection. See "Values".) ■ **4**: DECIMAL ■ **5**: TSTAMP (timestamp) For example, to provide an set of possible values, you set **Type** to **3**, and enter an array of values for this preference in **Values**. See "Values".

Table 6–1 (Cont.) Elements Used to Store a Subscriber Preference

Element	Description
String ID	Used for Localization. The ID in the **/string** class which would be associated with the localized string associated with the preference. Customer Center uses this information to display the preference name in a localized string form.
Default	The field containing the default value the preference is to be assigned
Values	An array list of values that the preference can assume. The Values array list is present only if the selection for **Type** is **ENUM**. See "Type".

3. For example, the following entry defines a new preference type called *Subscription Level* as the *tenth* preference for subscribers:

```
<SUBSCRIBER_PREFERENCES elem="10">
  <NAME>Subscription Level</NAME>
  <SUBSCRIBER_PREFERENCE_ID>10</SUBSCRIBER_PREFERENCE_ID>
  <STRING_ID>10</STRING_ID>
  <DEFAULT>Silver</DEFAULT>
  <TYPE>3</TYPE>
  <VALUES elem="0">
  <VALUE>Silver</VALUE>
  </VALUES>
  <VALUES elem="1">
  <VALUE>Gold</VALUE>
  </VALUES>
  <VALUES elem="2">
  <VALUE>Platinum</VALUE>
  </VALUES>
</SUBSCRIBER_PREFERENCES>
```

   In this example:

   - The **Name** of the preference is **Subscription Level**.
   - The subscriber preference ID for the language preference is **10**.
   - The string id for the localizing string is **10**.
   - The default value for the language preference is **Silver**.
   - The type of value is **3** (which is ENUM, and so an array of values follows).
   - The **Values** array lists the 3 possible subscription level selections: **Silver**, **Gold**, and **Platinum**

4. Save the **config_subscriber_preferences_map.xml** file.
5. Open the *BRM_Home***/apps/load_config/pin.conf** file in a text editor.
6. Add the following as the last entry:

```
- load_config validation_module libLoadValidTCFAAA LoadValidTelcoAAA_init
```

7. Save the **pin.conf** file.
8. Load the updated file by running the **load_config** utility:

```
load_config config_subscriber_preferences_map.xml
```

Important:

- The **load_config** utility requires a configuration (**pin.conf**) file.
- If you do not run the utility from the directory in which the configuration file is located, include the complete path to the file. For example,

  ```
  load_config BRM_Home/sys/data/config/config_subscriber_
  preferences_map.xml
  ```

For more information on the **load_config** utility, see *BRM Developer's Guide*.

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

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

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

7

How BRM Processes Prepaid AAA Requests

This chapter explains how the Oracle Communications Billing and Revenue Management (BRM) API processes prepaid authentication, authorization, and accounting (AAA) requests from external networks.

About Processing Prepaid AAA Requests

The main opcodes for processing prepaid AAA requests are the Services Framework AAA opcodes, which are generic framework opcodes that perform AAA operations common to all service types. Information is passed to these opcodes through either a service-specific manager, such as GSM AAA Manager, or a gateway application. For more information about the Services Framework, see "About Performing AAA for Prepaid Services".

You can use the BRM API to perform the following operations:

- Authenticate prepaid customers. See "How BRM Authenticates Prepaid Customers".
- Authorize customers to use a prepaid service. See "How BRM Authorizes Users to Access Prepaid Services".

  If you enabled in-session notifications, provide such notifications in the AAA responses to your network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".
- Reauthorize customers to continue a prepaid session. See "How BRM Reauthorizes Prepaid Services".

  If you enabled in-session notifications, provide such notifications in the AAA responses to your network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".
- Update prepaid sessions and then reauthorize usage. See "How BRM Updates and Reauthorizes Prepaid Sessions".

  If you enabled in-session notifications, provide such notifications in the AAA responses to your network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".
- Cancel authorization. See "How BRM Cancels Prepaid Service Authorizations".
- Manage prepaid sessions while they are in progress. See "How BRM Manages Prepaid Sessions".

How BRM Authenticates Prepaid Customers

For more information about prepaid authentication, see "About Authenticating Prepaid Customers".

The main opcode for authenticating prepaid customers is PCM_OP_TCF_AAA_AUTHENTICATE. You control whether BRM authenticates prepaid customers in Password Authentication Protocol (PAP) mode or Challenge-Handshake Authentication Protocol (CHAP) mode by passing the optional PIN_FLD_PASSWORD input flist field. If this field is passed in, BRM authenticates in PAP mode. If the field is not passed in, BRM authenticates in CHAP mode.

BRM authenticates users for prepaid services as follows:

1. PCM_OP_TCF_AAA_AUTHENTICATE calls the PCM_OP_ACT_FIND_VERIFY opcode to validate the user's identification. The MSID is used as the login field for PCM_OP_ACT_FIND_VERIFY.
2. PCM_OP_ACT_FIND_VERIFY calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, PCM_OP_ACT_FIND locates all the matching service instances of the mobile station ID (MSID) as the login, depending on whether you use a single-schema, a multischema, or an Oracle In-Memory Database (IMDB) Cache-enabled BRM system.
3. PCM_OP_ACT_FIND_VERIFY calls the PCM_OP_ACT_POL_SPEC_VERIFY policy opcode to retrieve the list of authentication checks to perform. The policy opcode returns the list of verification checks.
4. PCM_OP_ACT_FIND_VERIFY performs all checks specified by the policy opcode and returns to PCM_OP_TCF_AAA_AUTHENTICATE either success or failure.
5. PCM_OP_TCF_AAA_AUTHENTICATE returns to the caller either success or failure and, if CHAP authentication was used, the unencrypted password.

How BRM Authorizes Users to Access Prepaid Services

For more information about prepaid authorization, see "About Authorizing Prepaid Usage".

The main opcode for authorizing prepaid services is PCM_OP_TCF_AAA_AUTHORIZE. If your system is configured to handle in-session notifications from BRM (that is, the **piggyback** business parameter is enabled), this opcode appends in-session notifications to the responses it returns to the calling network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications" for more information.

BRM authorizes prepaid services as follows:

> **Note:** If you configure lightweight authorization, BRM authorizes prepaid services differently. See "How BRM Authorizes Users to Access Services When Lightweight Authorization Is Configured".

1. PCM_OP_TCF_AAA_AUTHORIZE calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

If the service type is not provided in the input flist, PCM_OP_ACT_FIND opcode locates all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
3. PCM_OP_TCF_AAA_AUTHORIZE uses the template to search for duplicate sessions. If the opcode finds a session with the same active session ID, authorization fails.
4. At the PREP_INPUT stage, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object to aggregate service-specific data. The helper opcode returns a service-specific input flist.
5. For policy-driven charging, checks to see if the request quota exceeds the available amount. See "Readjustment of Quota During Prepaid Authorization for Policy-Driven Charging". Also sets up an offer profile threshold breach notification if a threshold was breached in the offer profile for the service POID and account POID. See "About the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS Policy Opcode" in *BRM Setting Up Pricing and Rating*.
6. At the VALIDATE_LIFECYCLE stage, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object to validate the request if the service uses a custom life cycle. If validation succeeds, the authorization process continues. If validation fails, the request is denied.
7. PCM_OP_TCF_AAA_AUTHORIZE calls the PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES policy opcode to scale quantities for multi-RUM requests. See "Scaling Quantities for Prepaid Authorization Requests" for more information.
8. PCM_OP_TCF_AAA_AUTHORIZE passes the service-specific input flist to the PCM_OP_ACT_AUTHORIZE opcode.
9. PCM_OP_ACT_AUTHORIZE calls the PCM_OP_ACT_POL_PRE_AUTHORIZE policy opcode, which can be used to customize the authorization process.
10. If its input flist contains an **/active_session** type-only POID instead of an **/active_session** POID, PCM_OP_ACT_AUTHORIZE calls the PCM_OP_ASM_CREATE_ACTIVE_SESSION opcode to create an **/active_session** object (or a subclass of it).
11. PCM_OP_ACT_AUTHORIZE calls the PCM_OP_RESERVE_CREATE opcode to authorize the requested service and reserve resources for it. For multi-RUM requests, the input flist for PCM_OP_RESERVE_CREATE optionally includes a separate PIN_FLD_RUM_MAP array for each RUM. This array includes the minimum quantity for each RUM.

Note: If you have enabled the **Piggyback** business parameter to generate in-session notifications, PCM_OP_ACT_AUTHORIZE and PCM_OP_RESERVE_CREATE append credit threshold breach information to their responses. For more information on enabling **Piggyback**, see "Enabling In-Session Notifications".

To receive credit threshold breach information when you call PCM_OP_ACT_AUTHORIZE or PCM_OP_RESERVE_CREATE directly from **testnap** or a custom application, set the PIN_FLD_PIGGYBACK_FLAG in the input flist when you call the specific opcode. See "About Setting Up Custom Applications to Receive Credit Threshold Breach Information" for more information.

12. PCM_OP_RESERVE_CREATE does one of the following:

- **For an amount-based request**, compares the requested amount against the account's resources. Depending on the results of the comparison, PCM_OP_RESERVE_CREATE takes one of the actions listed in Table 7–1:

Table 7–1 Actions Taken by PCM_OP_RESERVE_CREATE

Comparison Result	PCM_OP_RESERVE_CREATE Action
Resources are greater than or equal to the requested amount.	■ Reserves the necessary resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_SUCCESS (**1**) in its output flist. ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.
Resources are less than the requested amount but greater than or equal to the input PIN_FLD_MIN_QUANTITY value.	■ Reserves the amount that can be covered by the account's resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**) in its output flist. ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.
Resources are less than the input PIN_FLD_MIN_QUANTITY value.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_NO_FUNDS (**4**) in its output flist. ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.

- **For a quantity-based request**, calls PCM_OP_ACT_USAGE in CALC_ONLY mode to determine whether the account's resources can cover the requested quantity or quantities. See "How Rating Works" in *BRM Setting Up Pricing and Rating*.
- If the full quantity or quantities cannot be covered, BRM determines the maximum resource amounts that can be reserved. See "Credit Limit Checks during Prepaid Authorization" for more information.

  If you enabled in-session notifications, PCM_OP_ACT_USAGE is called with PIN_FLD_PIGGGYBACK_FLAG set to **1** (enabled). PCM_OP_ACT_USAGE calculates the credit threshold breaches and returns any breach information in its output flist. See "How BRM Provides Credit Threshold Breach Notifications" for more information.

  Based on the results returned by PCM_OP_ACT_USAGE, PCM_OP_RESERVE_CREATE takes one of the actions listed in Table 7–2:

Table 7–2 Actions Taken by PCM_OP_RESERVE_CREATE

PCM_OP_ACT_USAGE Result	PCM_OP_RESERVE_CREATE Action
Single-RUM The account's resources cover the requested quantity. *Multi-RUM* The account's resources cover all requested quantities.	■ Reserves the necessary resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_SUCCESS (**1**) in its output flist. ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.
Single-RUM The account's resources do not cover the full requested quantity but do cover at least the minimum quantity for the RUM. *Multi-RUM* The account's resources cover the minimum quantity for all RUMs, but the full requested quantity cannot be covered for at least one RUM.	■ Reserves the available resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**). ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.
Single-RUM The available resources cover less than the minimum quantity for the RUM. *Multi-RUM* The available resources cover less than the minimum quantity for at least one RUM.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_RATED_QTY (**5**). ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.
Single-RUM The requested quantity is less than the minimum for the RUM. *Multi-RUM* Any requested quantity is less than the minimum quantity for that RUM.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INVALID_REQUESTED_QTY (**6**). ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.
Single-RUM No resources available. *Multi-RUM* No resources available for all RUMs.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_NO_FUNDS (**4**). ■ Returns the flist to PCM_OP_ACT_AUTHORIZE.

13. Based on the results passed in from PCM_OP_RESERVE_CREATE, PCM_OP_ACT_AUTHORIZE takes one of the actions listed in Table 7–3:

Table 7–3 Actions Taken by PCM_OP_ACT_AUTHORIZE

PCM_OP_RESERVE_CREATE Result	PCM_OP_ACT_AUTHORIZE Action
PIN_RESERVATION_SUCCESS	■ Calls the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode to add the reservation information to the **/active_session** object (or a subclass of it). ■ Sets PIN_FLD_RESULT to PIN_RESULT_PASS in its output flist.

Table 7–3 (Cont.) Actions Taken by PCM_OP_ACT_AUTHORIZE

PCM_OP_RESERVE_CREATE Result	PCM_OP_ACT_AUTHORIZE Action
PIN_RESERVATION_ INSUFFICIENT_FUNDS	■ Calls PCM_OP_ASM_UPDATE_ACTIVE_SESSION to add the reservation information to the **/active_session** object (or a subclass of it). ■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_PASS PIN_FLD_REASON to PIN_RESERVATION_ INSUFFICIENT_FUNDS (**3**)
PIN_RESERVATION_NO_ FUNDS	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_NO_ FUNDS (**4**)
PIN_RESERVATION_ INSUFFICIENT_RATED_QTY	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_ INSUFFICIENT_RATED_QTY (**5**)
PIN_RESERVATION_INVALID_ REQUESTED_QTY	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_ INVALID_REQUESTED_QTY (**6**)

When the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode is called with PIN_FLD_CREDIT_THRESHOLDS in its input flist, it updates the **/active_session** object with any credit threshold breach information.

14. If this is the first usage session of a resource whose validity period is based on first usage *and* the **SetFirstUsageInSession** business parameter is enabled, PCM_OP_ BAL_APPLY_MULTI_BAL_IMPACTS is called to set the start time of the resource's validity period.

    Note: By default, **SetFirstUsageInSession** is disabled, and the start time of the validity period is set at the end of the first usage session to the end time of that session. For more information, see "About Setting Resource Validity Periods Based on First Usage" in *BRM Setting Up Pricing and Rating*.

15. PCM_OP_TCF_AAA_AUTHORIZE calls the PCM_OP_ACT_POL_POST_ AUTHORIZE policy opcode to make any specified customizations to its output flist before returning the flist to the calling opcode. By default, the policy opcode drops the PIN_FLD_RESULTS field from the flist.

16. If the **Piggyback** business parameter is enabled, PCM_OP_ACT_AUTHORIZE calls the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode to set up the required in-session notifications. See "How BRM Provides In-Session Notifications" for more information.

Authorizing Multiple Services for a User with a Single Call

The PCM_OP_ACT_AUTHORIZE opcode, used during the BRM default prepaid authorization, authorizes a single service for a single customer at a time. The PCM_OP_ACT_MULTI_AUTHORIZE opcode offers the option of rating and authorizing multiple prepaid services with a single call. This opcode takes an array of services and a mode of operation as input and acts on them all within the same transaction. To take advantage of this option, you must write a custom program to replace the default BRM prepaid access authorization opcode flow using PCM_OP_ACT_MULTI_AUTHORIZE instead of PCM_OP_ACT_AUTHORIZE.

Note: BRM does not provide in-session notifications when it authorizes multiple services with a single call. An in-session notification is specific to a single service.

For details on the default prepaid authorization process, see "How BRM Authorizes Users to Access Prepaid Services".

For details on writing a custom program, see "About Customizing BRM" and "Writing a Custom Facilities Module" in *BRM Developer's Guide*.

Authorizing Multiple Services for a User without Reserving Resources

The PCM_OP_ACT_AUTHORIZE opcode, used during the BRM default prepaid authorization, reserves the resources that you pass in. The PCM_OP_ACT_MULTI_AUTHORIZE opcode offers the option of performing CALC_ONLY rating of the requested balances for each service passed in. To take advantage of this option, you must write a custom program to replace the default BRM prepaid access authorization opcode flow using PCM_OP_ACT_MULTI_AUTHORIZE instead of PCM_OP_ACT_AUTHORIZE.

Note: BRM does not provide in-session notifications when it authorizes multiple services with a single call. An in-session notification is specific to a single service.

For details on the default prepaid authorization process, see "How BRM Authorizes Users to Access Prepaid Services".

For details on writing a custom program, see "About Customizing BRM" and "Writing a Custom Facilities Module" in *BRM Developer's Guide*.

Authorizing Multiple Services for a User with Monetary Balances in a Non-BRM Database

The default BRM prepaid authorization works exclusively with accounts and balances contained in a BRM database. The PCM_OP_ACT_MULTI_AUTHORIZE opcode offers the option of performing CALC_ONLY rating based on monetary balances that you pass in, regardless of whether they originated in a BRM database.

Note: BRM does not provide in-session notifications when it authorizes multiple services with a single call. An in-session notification is specific to a single service.

PCM_OP_ACT_MULTI_AUTHORIZE calculates whether the account has sufficient resources for the requested service and returns the result. It treats monetary and non-monetary resources differently:

- For non-monetary resources, this opcode makes the calculation using the available balances in the BRM database.
- For monetary resources, this opcode makes the calculation using the available balances passed in.

To take advantage of this option, you must write a custom program to replace the default BRM prepaid access authorization opcode flow. Your application must use PCM_OP_ACT_MULTI_AUTHORIZE instead of PCM_OP_ACT_AUTHORIZE.

For details on the default prepaid authorization process, see "How BRM Authorizes Users to Access Prepaid Services".

For details on writing a custom program, see "About Customizing BRM" and "Writing a Custom Facilities Module" in *BRM Developer's Guide*.

How BRM Reauthorizes Prepaid Services

For more information about prepaid reauthorization, see "About Reauthorizing Prepaid Usage".

The main opcode for reauthorizing prepaid services is PCM_OP_TCF_AAA_REAUTHORIZE.

For policy-driven charging sessions, the network connectivity application provides the consumed quota in the reauthorization request. If BRM calculates that a threshold breach has occurred, it sets up an offer profile threshold breach event notification and calls the Payload Generator external module. (This application collects events, generates the data necessary to publish business events, and sends the data to the EAI-based (enterprise application integration) applications in your enterprise.) For more information, see "About the Data Synchronization Process" in *BRM Synchronization Queue Manager* and "Integrating BRM with Enterprise Applications" in *BRM Developer's Guide*.

For in-session notifications generated because you enabled the **Piggyback** business parameter, this opcode returns request and service-related information required by the network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".

BRM reauthorizes prepaid sessions as follows:

> **Note:** If you configure lightweight authorization, BRM reauthorizes prepaid services differently. See "How BRM Reauthorizes Prepaid Services When Lightweight Authorization Is Configured".

1. PCM_OP_TCF_AAA_REAUTHORIZE calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, the opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_REAUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
3. PCM_OP_TCF_AAA_REAUTHORIZE uses the template to search for the **/active_session** object.
   - If the object *is* found, the opcode calls the appropriate PREP_INPUT helper opcode to prepare a service-specific input flist and then passes the input flist to the PCM_OP_ACT_REAUTHORIZE opcode to reauthorize the call.
   - If the object *is not* found, the opcode calls the appropriate PREP_INPUT helper opcode to prepare a service-specific input flist and then passes the input flist to the PCM_OP_ACT_AUTHORIZE opcode to authorize the session with the given information. See "How BRM Authorizes Users to Access Prepaid Services".
4. PCM_OP_ACT_REAUTHORIZE determines whether the session's **/active_session** object (PIN_FLD_POID in the input flist), **/reservation_active** object (PIN_FLD_RESERVATION_OBJ in the input flist), and the **/config/reserve** object exist.
   - If the objects exist, the opcode performs the next step.
   - If the objects do not exist, the opcode calls PCM_OP_ACT_AUTHORIZE to perform a session authorization. See "How BRM Authorizes Users to Access Prepaid Services" for more information.

   Note: If Oracle IMDB Cache shuts down after a prepaid session begins, the session's **/active_session** and **/reservation_active** objects will no longer exist.
5. If the **/config/reserve** object contains a resource ID, this opcode retrieves the balances for all the resources associated with the resource ID, the associated offer profiles for the service POID and account POID. It reduces the consumed quota by the consumed reserved amount in the BALANCES array, and determines if the requested quota exceeds the amount available. If so, it readjusts the quota to the amount that is available.
6. PCM_OP_TCF_AAA_REAUTHORIZE calls the PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES policy opcode to scale quantities for multi-RUM requests. See "Scaling Quantities for Prepaid Authorization Requests" for more information for more information.
7. PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_ACT_POL_PRE_REAUTHORIZE policy opcode, which can be used to customize the reauthorization process.
8. PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_RESERVE_EXTEND opcode to reauthorize the requested service and extend its resource reservations. For multi-RUM requests, the input flist for PCM_OP_RESERVE_EXTEND optionally includes a separate PIN_FLD_RUM_MAP array for each RUM. This array includes the minimum quantity for each RUM.

Note: If you have enabled the **Piggyback** business parameter to generate in-session notifications, PCM_OP_ACT_REAUTHORIZE and PCM_OP_RESERVE_EXTEND append credit threshold breach information to their responses. For more information on enabling **Piggyback**, see "Enabling In-Session Notifications" for more information.

To receive credit threshold breach information when you call PCM_OP_ACT_REAUTHORIZE or PCM_OP_RESERVE_EXTEND directly from **testnap** or a custom application, set the PIN_FLD_PIGGYBACK_FLAG in the input flist when you call the specific opcode. See "About Setting Up Custom Applications to Receive Credit Threshold Breach Information" for more information.

9. PCM_OP_RESERVE_EXTEND does the following:

   - **For an amount-based request**, compares the requested amount against the account's resources. Depending on the results of the comparison, PCM_OP_RESERVE_EXTEND takes one of the actions listed in Table 7–4:

Table 7–4 Actions Taken by PCM_OP_RESERVE_EXTEND

Comparison Result	PCM_OP_RESERVE_EXTEND Action
Resources are greater than or equal to the requested amount.	■ Extends the appropriate reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_SUCCESS (**1**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Resources are less than the requested amount but greater than or equal to the input PIN_FLD_MIN_QUANTITY value.	■ Reserves the amount that can be covered by the account's resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Resources are less than the input PIN_FLD_MIN_QUANTITY value.	■ Does not extend any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_NO_FUNDS (**4**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.

   - **For a quantity-based request**,

     If the request is associated with a policy-driven charging session, the PCM_OP_RESERVE_EXTEND opcode updates the consumed reservation amount for the service in the **/reservation** and **/balance_group** objects.

     It calls PCM_OP_ACT_USAGE in CALC_ONLY mode to determine whether the account's resources can cover the requested quantity or quantities. See "How Rating Works" in *BRM Setting Up Pricing and Rating* for more information.

     If the full quantity or quantities cannot be covered, BRM determines the maximum resource amounts that can be reserved. See "Credit Limit Checks during Prepaid Authorization" for more information.

     If you have enabled the **Piggyback** business parameter to generate in-session notifications, PCM_OP_ACT_USAGE is called with PIN_FLD_

PIGGGYBACK_FLAG set to **1** (enabled). PCM_OP_ACT_USAGE calculates the credit threshold breaches and returns any breach information in its output flist. See "How BRM Provides Credit Threshold Breach Notifications" for more information.

Based on the results returned by PCM_OP_ACT_USAGE, PCM_OP_RESERVE_EXTEND takes one of the following actions listed in Table 7–5:

Table 7–5 Actions taken by PCM_OP_RESERVE_EXTEND

PCM_OP_ACT_USAGE Result	PCM_OP_RESERVE_EXTEND Action
Single-RUM The account's resources cover the requested quantity. *Multi-RUM* The account's resources cover all requested quantities.	■ Reserves the necessary resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_SUCCESS (**1**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM The account's resources cover more than the currently reserved quantity but do not cover the full requested quantity. *Multi-RUM* The account's resources cover more than the currently reserved quantity for all RUMs, but do not cover the full requested quantity for at least one RUM.	■ Reserves the available resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM The available resources cover less than the minimum quantity for the RUM. *Multi-RUM* The available resources cover less than the minimum quantity for at least one RUM.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_RATED_QTY (**5**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM The requested quantity is less than the minimum for the RUM. *Multi-RUM* Any requested quantity is less than the minimum quantity for that RUM.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INVALID_REQUESTED_QTY (**6**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM No resources available. *Multi-RUM* No resources available for all RUMs.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_NO_FUNDS (**4**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.

10. Based on the results passed in from PCM_OP_RESERVE_EXTEND, PCM_OP_ACT_REAUTHORIZE takes one of the actions listed in Table 7–6:

Table 7–6 Actions Taken by PCM_OP_ACT_REAUTHORIZE

PCM_OP_RESERVE_EXTEND Result	PCM_OP_ACT_REAUTHORIZE Action
PIN_RESERVATION_SUCCESS	■ Calls the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode to add the reservation information to the **/active_session** object (or a subclass of it). ■ Sets PIN_FLD_RESULT to PIN_RESULT_PASS in its output flist.
PIN_RESERVATION_INSUFFICIENT_FUNDS	■ Calls PCM_OP_ASM_UPDATE_ACTIVE_SESSION to add the reservation information to the **/active_session** object (or a subclass of it). When the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode receives PIN_FLD_CREDIT_THRESHOLDS in its input flist, it updates the **/active_session** object with any credit threshold breach information. ■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_PASS PIN_FLD_REASON to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**)
PIN_RESERVATION_NO_FUNDS	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON = PIN_RESERVATION_NO_FUNDS (**4**)
PIN_RESERVATION_INSUFFICIENT_RATED_QTY	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_INSUFFICIENT_RATED_QTY (**5**)
PIN_RESERVATION_INVALID_REQUESTED_QTY	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_INVALID_REQUESTED_QTY (**6**)

11. PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_ACT_POL_POST_REAUTHORIZE policy opcode to make any specified customizations to its output flist before returning the flist to the calling opcode. By default, the policy opcode drops the PIN_FLD_RESULTS field from the flist.
12. If the **Piggyback** business parameter is enabled, PCM_OP_TCF_AAA_REAUTHORIZE calls the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode to set up the required in-session notifications. See "How BRM Provides In-Session Notifications" for more information.

How BRM Updates and Reauthorizes Prepaid Sessions

The main opcode for updating and reauthorizing prepaid sessions is PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE.

For in-session notifications generated because you enabled the **Piggyback** business parameter, this opcode returns request-related and service-related information required by the network connectivity application. See "Providing In-Session

Notifications for Network Connectivity Applications".

BRM updates and reauthorizes prepaid sessions as follows:

1. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, PCM_OP_ACT_FIND opcode locates all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the PCM_OP_BAL_LOCK_RESERVATION_LIST opcode to find and lock the balance group's **/reservation_list** object. The opcode returns the POID of the **/reservation_list** object.

3. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode builds a search template for finding the **/active_session** object.

4. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE uses the search template to find the **/active_session** object.

5. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the PCM_OP_ACT_UPDATE_SESSION opcode to update the account balance in the **/active_session** object.

6. PCM_OP_ACT_UPDATE_SESSION determines whether to update or create the **/active_session** object by checking the PIN_FLD_POID input flist field.

   - If the field contains a *complete POID*, the opcode calls the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode to *update* the **/active_session** object's data and status. The opcode returns the POID of the updated **/active_session** object.
   - If the field contains a *POID type* only, the opcode calls the PCM_OP_ASM_CREATE_ACTIVE_SESSION opcode to *create* the **/active_session** object. The opcode returns the POID of the created **/active_session** object.

7. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE retrieves the updated balance from the **/active_session** object.

8. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the PCM_OP_ACT_REAUTHORIZE opcode to reauthorize the prepaid session.

9. For policy-driven charging, checks to see if the request quota exceeds the available amount. See "Readjustment of Quota During Prepaid Authorization for Policy-Driven Charging". Also sets up an offer profile threshold breach notification if a threshold was breached in the offer profile for the service POID and account POID. See "About the PCM_OP_BAL_POL_APPLY_MULTI_BAL_IMPACTS Policy Opcode" in *BRM Setting Up Pricing and Rating*.

10. PCM_OP_ACT_REAUTHORIZE determines whether the session's **/active_session** object (PIN_FLD_POID in the input flist) and **/reservation_active** object (PIN_FLD_RESERVATION_OBJ in the input flist) exist.

    If the objects do not exist, the opcode calls PCM_OP_ACT_AUTHORIZE to perform a session authorization. See "How BRM Authorizes Users to Access Prepaid Services".

Note: If Oracle IMDB Cache shuts down after a prepaid session begins, the session's **/active_session** and **/reservation_active** objects will no longer exist.

11. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES policy opcode to scale quantities for multi-RUM requests. See "Scaling Quantities for Prepaid Authorization Requests" for more information.
12. PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_ACT_POL_PRE_REAUTHORIZE policy opcode, which can be used to customize the reauthorization process.
13. PCM_OP_ACT_REAUTHORIZE calls PCM_OP_RESERVE_EXTEND to reauthorize the requested service and extend its resource reservations.

    For multi-RUM requests, the input flist for PCM_OP_RESERVE_EXTEND optionally includes a separate PIN_FLD_RUM_MAP array for each RUM. This array includes the minimum quantity for each RUM.

    Note: If you have enabled the **Piggyback** business parameter to generate in-session notifications, PCM_OP_ACT_REAUTHORIZE and PCM_OP_RESERVE_EXTEND append credit threshold breach information to their responses. For more information on enabling **Piggyback**, see "Enabling In-Session Notifications".

    To receive credit threshold breach information when you call PCM_OP_ACT_REAUTHORIZE or PCM_OP_RESERVE_EXTEND directly from **testnap** or a custom application, set the PIN_FLD_PIGGYBACK_FLAG in the input flist when you call the specific opcode. See "About Setting Up Custom Applications to Receive Credit Threshold Breach Information" for more information .
14. PCM_OP_RESERVE_EXTEND does one of the following:
    - **For an amount-based request**, compares the requested amount against the account's resources. Depending on the results of the comparison, PCM_OP_RESERVE_EXTEND takes one of the actions listed in Table 7–7:

Table 7–7 Actions Taken by PCM_OP_RESERVE_EXTEND

Comparison Result	PCM_OP_RESERVE_EXTEND Action
Resources are greater than or equal to the requested amount.	■ Extends the appropriate reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_SUCCESS (**1**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Resources are less than the requested amount but greater than or equal to the input PIN_FLD_MIN_QUANTITY value.	■ Reserves the amount that can be covered by the account's resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.

Table 7–7 (Cont.) Actions Taken by PCM_OP_RESERVE_EXTEND

Comparison Result	PCM_OP_RESERVE_EXTEND Action
Resources are less than the input PIN_FLD_MIN_QUANTITY value.	■ Does not extend any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_NO_FUNDS (**4**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.

- **For a quantity-based request**, calls PCM_OP_ACT_USAGE in CALC_ONLY mode to determine whether the account's resources can cover the requested quantity or quantities. See "How Rating Works" in *BRM Setting Up Pricing and Rating* for more information.

  If the full quantity or quantities cannot be covered, BRM determines the maximum resource amounts that can be reserved. See "Credit Limit Checks during Prepaid Authorization" for more information.

  If you enabled in-session notifications, PCM_OP_ACT_USAGE is called with PIN_FLD_PIGGYBACK_FLAG set to **1** (enabled). PCM_OP_ACT_USAGE calculates the credit threshold breaches and returns any breach information in its output flist. See "How BRM Provides Credit Threshold Breach Notifications" for more information.

  Based on the results returned by PCM_OP_ACT_USAGE, PCM_OP_RESERVE_EXTEND takes one of the actions listed in Table 7–8:

Table 7–8 Actions Taken by PCM_OP_RESERVE_EXTEND

PCM_OP_ACT_USAGE Result	PCM_OP_RESERVE_EXTEND Action
Single-RUM The account's resources cover the requested quantity. *Multi-RUM* The account's resources cover all requested quantities.	■ Reserves the necessary resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_SUCCESS (**1**) in its output flist. ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM The account's resources cover more than the currently reserved quantity but do not cover the full requested quantity. *Multi-RUM* The account's resources cover more than the currently reserved quantity for all RUMs, but do not cover the full requested quantity for at least one RUM.	■ Reserves the available resources. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM The available resources cover less than the minimum quantity for the RUM. *Multi-RUM* The available resources cover less than the minimum quantity for at least one RUM.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INSUFFICIENT_RATED_QTY (**5**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.

Table 7–8 (Cont.) Actions Taken by PCM_OP_RESERVE_EXTEND

PCM_OP_ACT_USAGE Result	PCM_OP_RESERVE_EXTEND Action
Single-RUM The requested quantity is less than the minimum for the RUM. *Multi-RUM* Any requested quantity is less than the minimum quantity for that RUM.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_INVALID_REQUESTED_QTY (**6**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.
Single-RUM No resources available. *Multi-RUM* No resources available for all RUMs.	■ Does not create any reservations. ■ Sets PIN_FLD_RESERVATION_ACTION to PIN_RESERVATION_NO_FUNDS (**4**). ■ Returns the flist to PCM_OP_ACT_REAUTHORIZE.

- For policy-driven charging sessions, the PCM_OP_RESERVE_EXTEND opcode updates the consumed reservation amount for the service in the **/reservation** and **/balance_group** objects.

15. Based on the results passed in from PCM_OP_RESERVE_EXTEND, PCM_OP_ACT_REAUTHORIZE takes one the actions in Table 7–9:

Table 7–9 Actions Taken by PCM_OP_ACT_REAUTHORIZE

PCM_OP_RESERVE_EXTEND Result	PCM_OP_ACT_REAUTHORIZE Action
PIN_RESERVATION_SUCCESS	■ Calls the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode to add the reservation information to the **/active_session** object (or a subclass of it). When the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode is called with PIN_FLD_CREDIT_THRESHOLDS in its input flist, it updates the **/active_session** object with any credit threshold breach information. ■ Sets PIN_FLD_RESULT to PIN_RESULT_PASS in its output flist.
PIN_RESERVATION_INSUFFICIENT_FUNDS	■ Calls PCM_OP_ASM_UPDATE_ACTIVE_SESSION to add the reservation information to the **/active_session** object (or a subclass of it). When the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode is called with PIN_FLD_CREDIT_THRESHOLDS in its input flist, it updates the **/active_session** object with any credit threshold breach information. ■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_PASS PIN_FLD_REASON to PIN_RESERVATION_INSUFFICIENT_FUNDS (**3**)
PIN_RESERVATION_NO_FUNDS	■ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON = PIN_RESERVATION_NO_FUNDS (**4**)

Table 7–9 (Cont.) Actions Taken by PCM_OP_ACT_REAUTHORIZE

PCM_OP_RESERVE_EXTEND Result	PCM_OP_ACT_REAUTHORIZE Action
PIN_RESERVATION_INSUFFICIENT_RATED_QTY	▪ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_INSUFFICIENT_RATED_QTY (**5**)
PIN_RESERVATION_INVALID_REQUESTED_QTY	▪ Sets the following in its output flist: PIN_FLD_RESULT to PIN_RESULT_FAIL PIN_FLD_REASON to PIN_RESERVATION_INVALID_REQUESTED_QTY (**6**)

16. If this is the first usage session of a resource whose validity period is based on first usage *and* the **SetFirstUsageInSession** business parameter is enabled, PCM_OP_BAL_APPLY_MULTI_BAL_IMPACTS is called to set the start time of the resource's validity period to the start time of the first usage session.

> **Note:** By default, **SetFirstUsageInSession** is disabled, and the start time of the validity period is set at the end of the first usage session to the end time of that session. For more information, see "About Setting Resource Validity Periods Based on First Usage" in *BRM Setting Up Pricing and Rating*.

17. PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_ACT_POL_POST_REAUTHORIZE policy opcode to make any specified customizations to its output flist before returning the flist to the calling opcode. By default, the policy opcode drops the PIN_FLD_RESULTS field from the flist.
18. At the POST_PROCESS stage, PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object.
19. PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE returns to the caller either success or failure.
20. If the **Piggyback** business parameter is enabled, PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode to set up the required in-session notifications. See "How BRM Provides In-Session Notifications" for more information.

How BRM Cancels Prepaid Service Authorizations

For more information about canceling prepaid authorization, see "About Canceling Authorizations for Prepaid Usage".

The main opcode for canceling authorization is PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION.

BRM cancels prepaid authorization as follows:

1. PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION calls PCM_OP_ACT_FIND to retrieve the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

If the service type is not provided in the input flist, the PCM_OP_ACT_FIND opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding the **/active_session** object.
3. PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION uses the search template to find the **/active_session** object to cancel. If the object is not found, the opcode generates an error.
4. PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION calls PCM_OP_ACT_CANCEL_AUTHORIZE with the **/active_session** POID, the list of reservations associated with the session, and the optional delete flag.
5. PCM_OP_ACT_CANCEL_AUTHORIZE calls PCM_OP_RESERVE_RELEASE to release any reserved resources.
6. PCM_OP_RESERVE_RELEASE calls the PCM_OP_RESERVE_POL_PRE_RELEASE policy opcode to perform custom validation.
7. PCM_OP_RESERVE_RELEASE updates the customer's reservation and account balances and then either saves or deletes the reservation object, depending on the value of the PIN_FLD_DELETED_FLAG input flist field:
   - If the flag is set to **1**, the opcode deletes the reservation object.
   - If the flag is set to any other value or is missing, the opcode saves the reservation object.

   For policy-driven charging sessions, PCM_OP_RESERVE_RELEASE clears the consumed reservation amount for the service from the **/balance_group** object.

   The opcode returns the **/account** POID and the list or reservation objects that were released.
8. PCM_OP_ACT_CANCEL_AUTHORIZE calls the PCM_OP_ASM_CLOSE_ACTIVE_SESSION opcode to cancel the active session.

   PCM_OP_ASM_CLOSE_ACTIVE_SESSION determines whether to delete the **/active_session** object by reading the PIN_FLD_STATUS_FLAG input flist field:
9. PCM_OP_ASM_CLOSE_ACTIVE_SESSION does the following:
   - If the PIN_FLD_STATUS_FLAG input flist field is:
     - **1**, this opcode deletes the **/active_session** object.
     - **0**, this opcode confirms that the object's status is **CREATED** or **UPDATED**, and then updates the object's status to **CANCELLED**.
   - Returns the POID of the **/active_session** object.
10. PCM_OP_ACT_CANCEL_AUTHORIZE returns to PCM_OP_TCF_AAA_CANCEL_AUTHORIZE the POID of the **/active_session** object.
11. PCM_OP_TCF_AAA_CANCEL_AUTHORIZE returns to the caller the **/account** POID and authorization ID.

How BRM Rates and Records Prepaid Activity Events

The main opcode for rating and recording prepaid activity events is PCM_OP_TCF_AAA_ACCOUNTING.

BRM rates and records prepaid activity events as follows:

1. PCM_OP_TCF_AAA_ACCOUNTING calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, the PCM_OP_ACT_FIND opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. PCM_OP_TCF_AAA_ACCOUNTING opens a master transaction and calls the PCM_OP_BAL_LOCK_RESERVATION_LIST opcode to lock the **/reservation_list** object.

3. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.

4. PCM_OP_TCF_AAA_ACCOUNTING uses the search template to look for duplicate session objects. If the opcode finds a session object with a matching authorization ID and a status of **Closed** or **Cancelled**, this is a duplicate request and the opcode returns with an error.

   Note: If it is operating in either the aggregate subsession mode or the deferred rate subsession mode, the opcode may find multiple **/active_session** objects that meet the criteria.

5. At the PREP_INPUT stage, PCM_OP_TCF_AAA_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object to aggregate service-specific data. The opcode returns a service-specific input flist.

   Note: When operating in either the aggregate subsession mode or the deferred rate subsession mode, the PREP_INPUT opcode returns the PIN_FLD_SESSION_INFO array with all of the subsession **/active_session** objects and master session information in the top level.

6. At the VALIDATE_LIFECYCLE stage, PCM_OP_TCF_AAA_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object to validate the request if the service uses a custom life cycle. If validation succeeds, the accounting process continues. If validation fails, the request is denied.

7. PCM_OP_TCF_AAA_ACCOUNTING passes the service-specific input flist to PCM_OP_ACT_ACTIVITY opcode.

   If the session includes a master session and one or more subsessions, PCM_OP_TCF_AAA_ACCOUNTING performs the following:

   a. Calls PCM_OP_ACT_ACTIVITY with information about the master session object. PCM_OP_ACT_ACTIVITY creates the event and calculates the balance impact for the activity.

b. For each subsession object passed in the PIN_FLD_SESSION_INFO array, calls PCM_OP_ACT_ACTIVITY with information about the subsession. The opcode also passes the **/event** POID of the master object and sets the subsession's status to one of the following:

   Closed, if the subsession mode is deferred rate mode or rate mode**.**

   Closed and unrated, if the subsession mode is aggregate mode.

8. PCM_OP_TCF_AAA_ACCOUNTING closes the master transaction.
9. PCM_OP_TCF_AAA_ACCOUNTING returns to the caller the **/event** POID, the authorization ID, the **/account** and **/service** POIDs, and the charge for the activity.

How BRM Refunds Charges for Prepaid Activity Events

The main opcode for refunding charges for prepaid activity events is PCM_OP_TCF_AAA_REFUND.

BRM refunds charges for prepaid activity events as follows:

1. PCM_OP_TCF_AAA_REFUND calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, the PCM_OP_ACT_FIND opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.
2. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_REFUND calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
3. PCM_OP_TCF_AAA_REFUND uses the search template to look for the **/event/session/*** and **/event/activity/*** objects.
4. PCM_OP_TCF_AAA_REFUND checks if the PIN_FLD_ADJUSTMENT_INFO array is present in the input flist.

   If the PIN_FLD_ADJUSTMENT_INFO array is present, PCM_OP_TCF_AAA_REFUND calls PCM_OP_AR_EVENT_ADJUSTMENT to perform event adjustment.

   If the PIN_FLD_ADJUSTMENT_INFO array is not present, PCM_OP_TCF_AAA_REFUND checks the PIN_FLD_CALL_DURATION field for the unused duration of time (in seconds) to be refunded.
5. PCM_OP_TCF_AAA_REFUND calls PCM_OP_ACT_USAGE in CALC_ONLY mode to determine the balance impacts.
6. PCM_OP_TCF_AAA_REFUND uses the balance impacts to prepare PIN_FLD_ADJUSTMENT_INFO.
7. PCM_OP_TCF_AAA_REFUND calls PCM_OP_AR_EVENT_ADJUSTMENT to perform event adjustment and then closes the master transaction.
8. PCM_OP_TCF_AAA_REFUND returns the **/event/notification/refund** event notification.

How BRM Manages Prepaid Sessions

For more information about managing prepaid sessions, see "About Accounting for Prepaid Sessions".

When a customer is authorized to use a prepaid service, BRM creates an **/active_session** object to record information about the customer's usage while the session is in progress. For example, the object stores the session's starting time, the associated reservation objects, and the amount the customer has currently consumed. When the session ends, BRM closes the **/active_session** object, records information about the completed session in an **/event/session** object, and then sends the session information to the rating process.

BRM tracks the **/active_session** object's life cycle by using the PIN_FLD_STATUS field. Active sessions can be set to one of the following states: CREATED, STARTED, UPDATED, RATED, or CLOSED.

How BRM Handles Quota Readjustment

When the requested quota is greater than the available amount, BRM readjusts the quota and sends that information to the network connectivity application (if the **piggyback** business parameter is enabled). See "Providing In-Session Notifications for Network Connectivity Applications" for more information.

How BRM Handles Offer Profile Threshold Breaches

For policy-driven sessions, BRM evaluates the requested quota in the incoming requests based on the configuration in the offer profile and the account details.

If BRM detects a breach of the policy threshold in the offer profile during the quota readjustment, it sends a threshold breach notification to the network policy controller. Such threshold breach notifications are sent for in-session and out-of-session occurrences of the threshold breach.

See "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.

How BRM Starts Prepaid Sessions

The main opcode for starting prepaid sessions is PCM_OP_TCF_AAA_START_ACCOUNTING. You use this opcode to record the session's starting time.

BRM starts a prepaid session as follows:

1. PCM_OP_TCF_AAA_START_ACCOUNTING calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, the PCM_OP_ACT_FIND opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_START_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding the **/active_session** object.

3. PCM_OP_TCF_AAA_START_ACCOUNTING uses the search template to find the **/active_session** object to update.

4. At the PREP_INPUT stage, PCM_OP_TCF_AAA_START_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object to aggregate service-specific data. The opcode returns a service-specific input flist.
5. PCM_OP_TCF_AAA_START_ACCOUNTING calls the PCM_OP_ACT_START_SESSION opcode with the input flist to start the prepaid session.
6. PCM_OP_ACT_START_SESSION either updates or creates the **/active_session** object, depending on the value of the PIN_FLD_POID input flist field:
   - If the field contains a *complete* POID, the opcode calls the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode to update the **/active_session** object and change its status from **Created** to **Started**.
   - If the field contains a *POID type* only, the opcode calls the PCM_OP_ASM_CREATE_ACTIVE_SESSION opcode to create the **/active_session** object, record the session start time, and set the object's status to **Created**.
7. PCM_OP_ACT_START_SESSION returns to PCM_OP_TCF_AAA_START_ACCOUNTING the POID of the **/active_session** object.
8. PCM_OP_TCF_AAA_START_ACCOUNTING returns to the caller the **/active_session** POID, **/account** POID, **/service** POID, and authorization ID.

How BRM Updates Prepaid Sessions

The main opcode for updating prepaid sessions is PCM_OP_TCF_AAA_UPDATE_ACCOUNTING. You use this opcode to find and update an existing **/active_session** object or, if one does not already exist, create an **/active_session** object.

BRM updates a prepaid session as follows:

1. PCM_OP_TCF_AAA_UPDATE_ACCOUNTING calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, the PCM_OP_ACT_FIND opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.
2. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_UPDATE_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
3. PCM_OP_TCF_AAA_UPDATE_ACCOUNTING uses the template to search for the **/active_session** object.
4. At the PREP_INPUT stage, PCM_OP_TCF_AAA_UPDATE_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object to aggregate service-specific data. The opcode returns a service-specific input flist.
5. PCM_OP_TCF_AAA_UPDATE_ACCOUNTING passes the data from the helper opcode to the PCM_OP_ACT_UPDATE_SESSION opcode.
6. PCM_OP_ACT_UPDATE_SESSION determines whether to *update* or *create* the **/active_session** object by checking the PIN_FLD_POID input flist field.
   - If the field contains a *complete POID*, the opcode calls the PCM_OP_ASM_UPDATE_ACTIVE_SESSION opcode to *update* the **/active_session** object's data and status. The opcode returns the POID of the updated **/active_session** object.

- If the field contains a *POID type* only, the opcode calls the PCM_OP_ASM_CREATE_ACTIVE_SESSION opcode to *create* the **/active_session** object. The opcode returns the POID of the created **/active_session** object.

7. PCM_OP_TCF_AAA_UPDATE_ACCOUNTING returns to the caller the **/active_session** POID, **/account** POID, **/service** POID, authorization ID, and current session balance.

How BRM Ends Prepaid Sessions

The main opcode for ending prepaid sessions is PCM_OP_TCF_AAA_STOP_ACCOUNTING. When a session ends, BRM performs the following tasks:

- Closes or deletes the **/active_session** object.
- Releases all reservations associated with the session.
- Rates the customer's usage.
- By default, if this is the first usage session, sets the validity start time to the end time of the session for validity periods based on first usage.

Note: The default behavior occurs when the **SetFirstUsageInSession** business parameter is disabled.

If **SetFirstUsageInSession** is enabled, the validity start time is set to the start time of the first usage session when BRM authorizes or updates and reauthorizes that session.

See "About Setting Resource Validity Periods Based on First Usage" in *BRM Setting Up Pricing and Rating*.

- Debits the customer's prepaid account balance.
- Stores information about the completed session in an **/event/session** object in the BRM database.
- Sends notification of any threshold breach to the network connectivity application.

BRM ends prepaid sessions as follows:

1. PCM_OP_TCF_AAA_STOP_ACCOUNTING calls PCM_OP_ACT_FIND to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.

   If the service type is not provided in the input flist, the PCM_OP_ACT_FIND opcode finds all the matching service instances of the MSID as the login, depending on whether you use a single-schema, a multischema, or an Oracle IMDB Cache-enabled BRM system.

2. PCM_OP_TCF_AAA_STOP_ACCOUNTING opens a master transaction and calls the PCM_OP_BAL_LOCK_RESERVATION_LIST opcode to lock the **/reservation_list** object.
3. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
4. PCM_OP_TCF_AAA_STOP_ACCOUNTING uses the search template to look for duplicate session objects. If the opcode finds a session object with a matching

authorization ID and a status of **Closed** or **Cancelled**, this is a duplicate request and the opcode returns with an error.

> **Note:** If it is operating in either the aggregate subsession mode or the deferred rate subsession mode, the opcode may find multiple **/active_session** objects that meet the criteria.

5. If an **/active_session** object is not found, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the PCM_OP_ASM_CREATE_ACTIVE_SESSION opcode to create an **/active_session** object with the information from the input flist.
6. At the PREP_INPUT stage, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object to aggregate service-specific data. The opcode returns a service-specific input flist.

   > **Note:** When operating in either the aggregate subsession mode or the deferred rate subsession mode, the PREP_INPUT opcode returns the PIN_FLD_SESSION_INFO array and the master session data in the top level of the flist.

7. (Direct debit mode Only) At the VALIDATE_LIFECYCLE stage, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the helper opcode specified in the **/config/opcodemap/tcf** object to validate the request if the service uses a custom life cycle. If validation succeeds, the stop accounting process continues. If validation fails, the request is denied.
8. PCM_OP_TCF_AAA_STOP_ACCOUNTING passes the service-specific flist to the PCM_OP_ACT_END_SESSION opcode with information about the session.

   If the session includes a master session and one or more subsessions, PCM_OP_TCF_AAA_ACCOUNTING performs the following, depending on the subsession mode:

   - In aggregate mode, the opcode cleans up all of the subsessions and then calls the PCM_OP_ACT_END_SESSION opcode to rate and record the event for the master session.
   - In deferred rate mode, the opcode rates the master session first by calling PCM_OP_ACT_END_SESSION and retrieving the event POID. Then, the opcode rates all of the subsessions by calling the PCM_OP_ACT_END_SESSION opcode and passing the master session POID.
   - In rate mode, the opcode calls PCM_OP_ACT_END_SESSION to rate and record the session.

9. PCM_OP_TCF_AAA_ACCOUNTING confirms that the master session and all subsessions have ended and have been processed.

   > **Note:** You can configure the opcode to skip this step by using the Services Framework AAA **pin.conf** file. For more information, see "Ensuring That All Subsessions Have Stopped before Closing the Master Session".

10. PCM_OP_TCF_AAA_ACCOUNTING closes the master session and subsessions:

- In aggregation mode, the opcode sets the master session to **Closed** and all subsessions to **Closed and Unrated**.
- In deferred rate mode, the opcode sets the master session and all subsessions to **Closed**.
- In rate mode, the opcode sets the master session and all subsessions to **Closed**.

11. PCM_OP_TCF_AAA_STOP_ACCOUNTING returns to the caller the **/event/session** POID, the **/account** POID, the **/service** POID, and the rated amount to the calling application.
12. If the **Piggyback** business parameter is enabled, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the PCM_OP_TCF_AAA_POL_POST_PROCESS policy opcode to set up the required in-session notifications. See "How BRM Provides In-Session Notifications" for more information.

How BRM Closes Prepaid Sessions When the External Network Shuts Down

The main opcode for closing active sessions when the external network shuts down is PCM_OP_TCF_AAA_ACCOUNTING_OFF. You can control whether BRM rates newly created active sessions before closing them by using the PIN_FLD_ACC_FLAG input flist field.

BRM closes sessions when the network shuts down as follows:

1. At the ACC_ON_OFF_SEARCH stage, PCM_OP_TCF_AAA_ACCOUNTING_OFF calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding **/active_session** objects.
2. PCM_OP_TCF_AAA_ACCOUNTING_OFF uses the search template to find all session objects that meet the criteria.
3. PCM_OP_TCF_AAA_ACCOUNTING_OFF performs one of the following for each session object, depending on the object's status:
   - If the object's PIN_FLD_STATUS field is set to **CREATED** *and* the PIN_FLD_ACC_FLAG input flist field is set to **TRUE**, calls the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode to end the session and rate any usage.
   - If the object's PIN_FLD_STATUS field is set to **CREATED** *and* the PIN_FLD_ACC_FLAG input flist field is set to **FALSE**, calls the PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION opcode to cancel the authorization.
   - If the object's PIN_FLD_STATUS field is set to **STARTED** or **UPDATED**, calls PCM_OP_TCF_AAA_STOP_ACCOUNTING to end the session and rate any usage.
4. PCM_OP_TCF_AAA_ACCOUNTING_OFF returns the POID of each closed object.

How BRM Closes Prepaid Sessions When the External Network Restarts

The main opcode for closing active sessions when the external network restarts is PCM_OP_TCF_AAA_ACCOUNTING_ON.

You can control whether BRM rates newly created active sessions before closing them by using the PIN_FLD_ACC_FLAG input flist field.

BRM closes sessions when the network shuts down as follows:

1. At the ACC_ON_OFF_SEARCH stage, PCM_OP_TCF_AAA_ACCOUNTING_ON calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding **/active_session** objects.
2. PCM_OP_TCF_AAA_ACCOUNTING_ON uses the search template to find all session objects that meet the criteria.
3. PCM_OP_TCF_AAA_ACCOUNTING_ON performs one of the following for each session object, depending on the object's status:
   - If the object's PIN_FLD_STATUS field is set to **CREATED** *and* the PIN_FLD_ACC_FLAG input flist field is set to **TRUE**, calls the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode to end the session and rate any usage.
   - If the object's PIN_FLD_STATUS field is set to **CREATED** *and* the PIN_FLD_ACC_FLAG input flist field is set to **FALSE**, calls the PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION opcode to cancel the authorization.
   - If the object's PIN_FLD_STATUS field is set to **STARTED** or **UPDATED**, calls PCM_OP_TCF_AAA_STOP_ACCOUNTING to end the session and rate any usage.
4. PCM_OP_TCF_AAA_ACCOUNTING_ON returns the POID of each closed object.

Scaling Quantities for Prepaid Authorization Requests

You use the PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES policy opcode to scale multi-RUM quantities for prepaid authorization and reauthorization requests. This opcode is called by the PCM_OP_TCF_AAA_AUTHORIZE opcode or the PCM_OP_TCF_AAA_REAUTHORIZE opcode during prepaid authorization and reauthorization for multi-RUM quantities.

For initial authorization requests, the PCM_OP_ACT_POL_SCALE_MULTI_RUM_QUANTITIES opcode scales quantities based on ratios that you specify in the **/config/reserve** object for the appropriate service. See "Specifying Default Authorization and Reauthorization Values" for more information.

For reauthorization, the opcode can use the actual usage for the session to determine the ratio of the RUMs to be reauthorized. If its input flist includes information about used quantities for each RUM, the opcode automatically calculates the scaling ratio based on the usage. If usage information is not included in the input flist, the opcode continues to use the configured ratio.

Scaling is based on the relationship of each RUM to the primary RUM. The opcode identifies the primary RUM by the presence of the PIN_FLD_IS_PRIMARY_RUM field in the PIN_FLD_QUANTITIES array for the RUM. There can be only one primary RUM in a request.

The formula used for scaling each non-primary RUM is:

Net quantity = Requested quantity of primary RUM * Ratio (static or dynamic)

where:

- The static ratio is the ratio configured in the **/config/reserve** object.
- The dynamic ratio is calculated for each RUM based on usage information:

  Ratio = Quantity used for the RUM/Quantity used of primary RUM

You can modify this policy opcode to implement customized scaling behavior. For example, you can change the default formulas used for scaling.

Credit Limit Checks during Prepaid Authorization

When BRM authorizes prepaid usage, it performs a credit limit check to determine whether the customer has sufficient resources available to authorize the requested quantity. See "About Determining whether there Are Sufficient Resources" for an overview.

The credit limit check takes place in one of two possible locations:

- If the event is rated by a single RUM and is not eligible for discounts or charge sharing, the credit check is performed by rating opcodes during real-time rating. See "About Credit Limit Checks by Rating Opcodes" for more information.
- If an event is rated by multiple RUMs or is eligible for discounts or charge sharing, the credit limit check is performed by the FCT_CreditLimitCheck module in the real-time discounting pipeline. See "About Credit Limit Checks in the Real-Time Discounting Pipeline" for more information.

About Credit Limit Checks by Rating Opcodes

For events that are rated by a single RUM and are not eligible for discounts or charge sharing, credit limit checks are performed by the PCM_OP_RATE_EVENT opcode in CALC_ONLY mode during real-time rating. When the opcode rates the event in CALC_ONLY mode, it compares the rated amount against the customer's current prepaid balance, less any active reservations. If the customer's available resources are greater than the rated amount, the requested quantity is authorized and resources are reserved. See "How Rating Works" in *BRM Setting Up Pricing and Rating*.

If the customer's resources cannot allocate the entire requested quantity, PCM_OP_RATE_EVENT reserves the quantity that can be authorized and returns the quantity that cannot be reserved as unrated quantity.

About Credit Limit Checks in the Real-Time Discounting Pipeline

BRM performs credit limit checks in the real-time discounting pipeline under the following circumstances:

- The event being authorized is rated by multiple RUMs. For example, an authorization for downloading data could be rated by both duration and volume. In this case, the user's resources must be sufficient for both RUMs.
- The event being authorized is eligible for discounts, including discount sharing, charge sharing, free minutes, loyalty points, and so on. Discounts need to be applied prior to checking the credit limit.

In either of these situations, real-time rating rates the event but does not perform a credit check. It sets the CREDIT_LIMIT_CHECK_FLAG and passes information about the event to the real-time discounting pipeline, which performs the credit check by using the FCT_CreditLimitCheck module. See *BRM Configuring Pipeline Rating and Discounting* for more information.

FCT_CreditLimitCheck performs its credit limit check in two stages:

- In the first stage, this module estimates the cost of usage, factoring in all discount and charge sharing amounts, and compares it against the customer's available resources. If more than one RUM applies, the estimate includes all RUMs. If the check fails, the module moves to the second stage. If the check passes, the module skips the second stage.
- If the first stage of the credit limit check fails, the module determines the maximum quantity or quantities that can be authorized given the customer's

resources. The module determines the maximum authorized quantities for single-RUM with discounts and multiple-RUM scenarios. See "Calculating Maximum Quantities for Single-RUM Authorization Requests with Discounts" and "Calculating Maximum Quantities for Multi-RUM Authorization Requests" for more information.

FCT_CreditLimitCheck performs credit limit checks as follows:

1. Determines whether to perform credit limit checking by reading the DETAIL.CREDIT_LIMIT_CHECK field in the event data record (EDR.)
   - If CREDIT_LIMIT_CHECK is set to **1**, the module proceeds to the next step.
   - If CREDIT_LIMIT_CHECK is set to **0**, the module skips the credit limit check and sets the DETAIL.CREDIT_LIMIT_CHECK_RESULT field to **1**.
2. Performs a simple credit limit check by comparing the net impact on each resource in each balance group to the balances. See "Determining Whether there Are Sufficient Resources" for more information.
   - If resources are sufficient to authorize, the module sets the DETAIL.CREDIT_LIMIT_CHECK_RESULT field to **1** and sets the DETAIL.UNRATED_QUANTITY field to **0**.
   - If any balances are insufficient, the module proceeds to the next step.
3. Determines the maximum quantity or quantities that can be authorized. See "Calculating Maximum Quantities for Single-RUM Authorization Requests with Discounts" and "Calculating Maximum Quantities for Multi-RUM Authorization Requests" for more information.
4. Sets the DETAIL.CREDIT_LIMIT_CHECK_RESULT field to **0** and adds information about quantities and resources to the EDR.

   For multi-RUM requests, separate charge and discount packets are created for each RUM. The DETAIL.ASS_CBD.RM.UNRATED_QUANTITY field is set to the quantity that cannot be rated for each RUM.

The data from the EDR is passed back to the reservation opcodes, which reserve the appropriate resources.

Determining Whether there Are Sufficient Resources

When an authorization request is made, FCT_CreditLimitCheck determines whether there are sufficient resources to authorize the requested quantity by comparing the balance impact of all the charge packets for each RUM with the balance of each resource.

The charge packets are created by the FCT_Discount module based on the QUANTITY_FROM and QUANTITY_TO values in the charge and discount packets. FCT_Discount breaks the requested quantity into linear segments such that each charge packet has a single net rate, after applying the discounts and charge shares.

For example, if rate R1 applies for the first 50 minutes, rate R2 applies for all subsequent usage, and discount D1 applies for the first 25 minutes, there would be three segments for a 100-minute authorization, as shown in Table 7–10:

Table 7–10 Calculating Usage by Rates

Segment	Rate	Discount	Net Rate
0 – 25	R1	D1	R1 - D1
25 – 50	R1	none	R1

Table 7–10 (Cont.) Calculating Usage by Rates

Segment	Rate	Discount	Net Rate
50 – 100	R2	none	R2

Using the segment information added to the EDR by FCT_Discount, FCT_CreditLimitCheck:

1. Sequentially checks each charge packet to determine whether the customer has enough resources to allocate for the net impact of the charge after the discount for that segment. The resources that can be reserved are added to the quantity to be authorized.
2. When it reaches a charge packet that cannot be authorized fully by the customer's resources, calculates the portion of the charge packet that can be authorized. This prorated quantity is added to the quantity to be authorized.
3. Sums the quantities for each packet and returns that total quantity for authorization.

The following example shows how credit limit check is performed for non-currency resource balances.

Suppose that a customer wants to use a service that costs $1.00 per minute. The customer's current prepaid balance is $20 and 10 free minutes resource balance. The customer is eligible for 10% discount on the first 10 minutes and 20% discount after that. The authorization request is for 30 minutes.

FCT_Discount breaks the requested quantity into three segments, as shown in Table 7–11:

Table 7–11 Request Segmentation by FCT_Discount

Time	Rate	Discount	Net Rate
0 – 10	$1.00/min.	–100%	$0/min.
10 – 20	$1.00/min.	–10%	$0.90/min.
20 – 30	$1.00/min.	–20%	$0.80/min.

Using the segment information, FCT_CreditLimitCheck:

1. Sequentially checks each charge packet to determine whether the customer's resource balance can authorize it.
   - The first charge packet (10 minutes) is $0. It consumes from the free resource buckets first, leaving 0 free minutes and $20 prepaid balance.
   - The second charge packet (10 minutes) is $9, leaving a $11 prepaid balance.
   - The third charge packet (10 minutes) is $8, leaving a $3 prepaid balance.
2. Sums the quantities for each charge packet and returns that total quantity for authorization. The total quantity authorized is 30 minutes at a cost of $17.00.

The following example shows how credit limit check is performed for currency resource balances.

Suppose that a customer wants to use a service that costs $5.00 per minute. The customer's current prepaid balance is $30 and a free monetary credit of $20. The customer is eligible for a 10% discount on the first $10 charge and a 20% discount on all charges after that. The authorization request is for 10 minutes.

FCT_Discount converts the equivalent charge to the equivalent call duration and breaks the requested quantity into three segments, as shown in Table 7–12:

Table 7–12 Request Segmentation for Equivalent Charges by FCT_Discount

Time	Rate	Discount	Net Rate
0 – 4	$5.00/min.	–100%	$0.00/min.
4 – 6	$5.00/min.	–10%	$4.50/min.
6 – 10	$5.00/min.	–20%	$4.00/min.

Using the segment information, FCT_CreditLimitCheck:

1. Sequentially checks each charge packet to determine whether the customer has resources to authorize it.
   - The first charge packet (4 minutes) is $0. It consumes from the free resource buckets first, leaving a $0 free monetary credit balance and $30 prepaid balance.
   - The second charge packet (2 minutes) is $9, leaving a $21 prepaid balance.
   - The third charge packet (4 minutes) is $16, leaving a $5 prepaid balance.
2. Sums the quantities for each segment and returns that total quantity for authorization. The total quantity authorized is 10 minutes at a cost of $25.00.

Calculating Maximum Quantities for Single-RUM Authorization Requests with Discounts

When an initial credit limit check fails for a single-RUM authorization request, FCT_CreditLimitCheck determines the maximum quantity that can be authorized by sequentially checking segments of the request.

Suppose that a customer wants to use a service that costs $1.00 per minute for the first 40 minutes and $0.50 per minute after that. The customer is eligible for a 20% discount on the first 10 minutes and a 40% discount after that. The customer's current balance is $38 and the authorization request is for 100 minutes.

FCT_Discount breaks the requested quantity into multiple charge and discount packets based on the rates and discounts that apply. Table 7–13 shows the three segments:

Table 7–13 Request Segmentation by FCT_Discount

Time	Rate	Discount	Net Rate
0 – 10	$1.00/min.	–20%	$0.80/min.
10 – 40	$1.00/min.	–40%	$0.60/min.
40 – 100	$0.50/min.	–40%	$0.30/min.

The initial check in FCT_CreditLimitCheck fails because the cost for 100 minutes ($44) is greater than the customer's resources ($38). The module then does the following to determine the maximum quantity that can be authorized for the session:

1. Sequentially checks each segment to determine whether the customer has resources to authorize it.
   - The first segment costs less than the balance of $38. Its 10 minutes can be authorized at a cost of $8, leaving a $30 balance.

- The second segment costs less than the remaining balance of $30. Its 30 minutes can be authorized at a cost of $18, leaving a $12 balance.
- The third segment costs more than the remaining balance of $12, so the segment must be prorated. The net cost per minute is $0.30, so 40 minutes can be authorized for the remaining balance of $12.

2. Sums the quantities for each segment and returns that total quantity for authorization. The total that can be authorized is 80 minutes at a cost of $38.00.

Calculating Maximum Quantities for Multi-RUM Authorization Requests

If an initial credit limit check fails for a multi-RUM authorization request, FCT_CreditLimitCheck determines the maximum quantity that can be authorized by checking incrementally smaller or larger quantities until the maximums are reached. The maximum is based on the total charge of the charge packets of all RUMs.

The module uses a method similar to a binary search to determine the size of each increment to check, stopping when the increment becomes smaller than the step size for one of the resources.

Note: Currently, the step size is always 1.

To find the maximum quantities of each resource, FCT_CreditLimitCheck:

1. Divides the originally requested quantities in half.
2. Compares the cost of the new quantities to the customer's balance, then does one of the following:
   - If the cost is *lower* than the customer's balance, increases the requested quantities. Each quantity is increased by half of the difference between it and the originally requested quantity.
   - If the cost is *greater* than the customer's balance, decreases the requested quantities. Each quantity is reduced by half of the difference between it and zero.
   - If the cost of the new quantities is a match for the customer's balance, returns those quantities for authorization.
3. Compares the combined cost of the quantities determined in the previous step to the customer's balance, then does one of the following:
   - If the cost is *lower* than the customer's balance, increases the requested quantities. Each quantity is increased by half of the difference between it and the original quantity *or* the previously requested quantity, whichever is smaller.
   - If the cost is *greater* than the customer's balance, decreases the requested quantities. Each quantity is reduced by half of the difference between the current value and zero *or* the previously requested quantity, whichever is greater.
   - If the cost of the new quantities is a match for the customer's balance, returns those quantities for authorization.
4. Repeats step 3 until dividing the remaining quantities in half results in an increment that is smaller than the step size for at least one resource.
5. Returns the last successfully checked quantities for authorization.

For example, suppose that a customer requests authorization for a GPRS session with a 20-minute duration and 40 megabytes of data. Minutes are charged at $0.40 per minute and data is charged at $0.50 per megabyte. The customer has a $20 balance.

The initial credit limit check fails because the session cost ($28) is greater than the customer's balance ($20).

FCT_CreditLimitCheck then does the following to determine the maximum quantities that can be authorized for the session:

1. Divides the initially requested quantities for each RUM in half and checks them as shown in Table 7–14.

Table 7–14 Dividing and Checking Request Quantities

RUM 1	RUM 1	RUM 2	RUM 2	None	None	None
Time	Cost	Volume	Cost	Total	< or > balance	Result
10 min.	$4.00	20 MB	$10.00	$14.00	<	Try larger

2. The total cost ($14) is less than the customer's balance ($20), so the module increases the quantity of each resource by half of the amount between it and the original quantity, then checks the resulting quantities as shown in Table 7–15.

Table 7–15 Adjusting Request Quantities

RUM 1	RUM 1	RUM 2	RUM 2	RUM 2	RUM 2	RUM 2
Time	Cost	Volume	Cost	Total	< or > balance	Result
15 min.	$6.00	30 MB	$15.00	$21.00	>	Try smaller

3. The total cost ($21) is greater than the customer's balance, so the module reduces the quantity of each RUM by the half the difference between the first and second attempts, then checks the resulting quantities as shown in Table 7–16.

Table 7–16 Further Adjustment of Request Quantities

RUM 1	RUM 1	RUM 2	RUM 2	RUM 2	RUM 2	RUM 2
Time	Cost	Volume	Cost	Total	< or > balance	Result
12.5 min.	$5.00	25 MB	$12.50	$17.50	<	Try larger

4. The total cost ($17.50) is less than the customer's balance, so the module increases the quantity of each RUM by half the difference between the second and third attempts, then checks the resulting quantities as shown in Table 7–17.

Table 7–17 Final Adjustment of Request Quantities

RUM 1	RUM 1	RUM 2	RUM 2	RUM 2	RUM 2	RUM 2
Time	Cost	Volume	Cost	Total	< or > balance	Result
13.75 min.	$5.50	27.5 MB	$13.75	$19.25	<	Stop

5. The cost ($19.25) is less than the customer's balance, but the module cannot try a larger quantity because the minimum step size for minutes is 1. If rounding is enabled, the module returns 14 minutes and 28 megabytes for authorization. If rounding is not enabled, the module returns 13.75 minutes and 27.5 megabytes for authorization. See "Enabling Rounding for Maximum Quantity Results" for more information.

Enabling Rounding for Maximum Quantity Results

You can choose to have FCT_CreditLimitCheck round up the results of its maximum quantity calculations to the nearest whole number. For example, if the module calculates a maximum quantity of 74.4, rounding would change the quantity to 75.

> **Note:** Rounded quantities may exceed the customer's available resources by a small amount. The customer will not be charged for the fraction used to round up the quantity.

Rounding is disabled by default. You enable rounding by setting **RoundUpRequestQuantity** to **True** in the module's registry entry. See "FCT_ CreditLimitCheck" in *BRM Configuring Pipeline Rating and Discounting* for more information.

> **Note:** Windows users: To display decimals to the desired precision, ensure that your regional settings are properly defined.

Special Cases for Calculating Maximum Authorizations

Because of the way that BRM calculates maximum authorizations, keep the following cases in mind:

- Threshold Discounts
- Sponsor-Limited Authorizations
- Conditional Discounts

Threshold Discounts

Depending on the discount configuration, authorizations for sessions that involve threshold discounts can be incorrect. This occurs because the discount threshold may be triggered by the initial quantity that BRM tries, but not by the quantity actually authorized. BRM cannot know in advance what the actual authorized quantity will be, so it cannot be sure whether the threshold discount applies.

For example, assume that a service is charged at $1.00 minute and is eligible for a threshold discount of $0.50 for calls over 50 minutes. If the threshold value is reached, the discount applies to the entire session.

Suppose the customer's balance is $20. The network attempts an authorization of 100 minutes. Because of its length, the session is eligible for the discount of $0.50 per minute, but the cost ($50) is greater than the customer's balance ($20). As a result, the initial credit limit check by FCT_CreditLimitCheck fails.

The module then calculates the maximum quantity that can be authorized. It bases the calculation on the net cost after the discount, which yields an authorization of 40 minutes.

However, the 40 minutes do not qualify for the $0.50-per-minute discount because it is below the 50-minute threshold. When the call is actually rated, the non-discounted rate applies. This results in a charge of $40, which is larger than the customer's balance.

Sponsor-Limited Authorizations

When an authorization request includes charge sharing, the charge sharing sponsor's balance is factored into credit limit checks. As a result, the sponsor's balance rather than the customer's balance may be the factor that determines the maximum quantity that can be authorized.

For example, assume a customer has a service that is charged at $1.00 per minute and is eligible for a charge share of 40%. If the customer has a balance of $66 and the sponsor a balance of $32, an authorization request for 100 minutes will fail. The customer's balance is more than the amount required to authorize 60% of the charge, but the sponsor's balance of $32 cannot authorize 40% of $100.

FCT_CreditLimitCheck then calculates the maximum quantity that can be authorized. That maximum is reached when *either* the customer's or the sponsor's balance is reached. In this case, the sponsor's balance is the limiting factor:

- For the customer, the net rate is $0.60 per minute. The maximum that could be authorized for the customer's $66 balance is 110 minutes, more than the requested quantity.
- For the sponsor, the net rate is $0.40 per minute. The maximum that can be authorized for the sponsor's balance of $32 is 80 minutes.

FCT_CreditLimitCheck would return 80 minutes to be authorized. The sponsor's total balance of $32 would be reserved as well as $48 of the customer's balance.

Conditional Discounts

Discounts that are based on conditions, such as monthly usage totals, cannot be reliably authorized. This occurs because there is no way to guarantee in advance that the condition will be met.

For example, suppose that a customer is eligible for a discount starting when his monthly usage total reaches a certain level. If the authorization request includes enough usage to exceed the specified level, the customer will be authorized for a larger total than would be possible without the discount. However, when the customer's actual usage is rated, it may not be sufficient to trigger the discount condition. In this case, the customer's actual usage may exceed what can be authorized by his available resources.

Readjustment of Quota During Prepaid Authorization for Policy-Driven Charging

During prepaid usage authorization, when BRM receives a request to authorize or to update and reauthorize the quota for a session, it readjusts the requested quota based

on the current balance, used reservation across all the parallel sessions, and the nearest threshold configured in the offer profile.

At the PREP_INPUT stage of the AAA flow, if **/config/reserve** object contains a resource id, BRM performs the following operations:

- Obtain the balances for all the resources by calling the PCM_OP_BAL_GET_PREPAID_BALANCES opcode.
- Retrieve all the offer profiles associated with the service POID and account POID by calling the PCM_OP_OFFER_PROFILE_GET_OFFER_PROFILE opcode.
- Calculate the consumed quota. It reduces the consumed quota obtained from the active session object by the CONSUMED_RESERVED _AMOUNT in the BALANCES array for the reservation.
- Check if any threshold is crossed by calling the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD opcode and passing offer profiles, service object, resource id, and the updated consumed quota calculated above.

  See "About the PCM_OP_OFFER_PROFILE_CHECK_POLICY_THRESHOLD Opcode" in *BRM Setting Up Pricing and Rating* for more information.
- If the requested quota exceeds the available amount, authorize the request for just the available amount.

For example, you configure an offer profile for a non-currency resource such as *Megabytes (MB) Used* with specific thresholds (at 140MB, 160 MB, and so on).

- The current balance on an account for this resource is 80 MB and the consumed reserved amount (across parallel sessions, iPhone, video, and computer) is 35 MB.

  The first threshold in the offer profile is 140 MB making the allowable reservation quota 25 MB (140-115).
- When BRM receives a request to authorize for that resource and the requested amount exceeds 25 MB, BRM sets the allowable reservation quota for the session at 25 MB.

  The session is initiated.
- BRM receives an update and reauthorize request. The consumed quota input in that request is 25 MB.
- The consumed amount is 140 MB (80 + 35 + 25). The next threshold in the offer profile is 160. This means that the allowable quota is 20 MB.
- BRM calculates the consumed amount as 140 MB (80 + 35 + 25). BRM uses the next threshold in the offer profile and calculates 20 MB as the allowable quota (160-140).
- If the update and authorize request is for an amount greater than 20 MB, BRM readjusts the allowance to 20 MB.

For more information on:

- Authorizing a request, see "How BRM Starts Prepaid Sessions".
- Updating and Reauthorizing a request, see "How BRM Updates Prepaid Sessions".
- Policy-Driven charging, see "Policy-Driven Charging" in *BRM Setting Up Pricing and Rating*.

8

Using Lightweight Authorization

This chapter describes the Oracle Communications Billing and Revenue Management (BRM) lightweight authorization feature. This feature enables you to override the resource availability, reservation amount, and scaled delay time for the service of a particular account.

Before you read this chapter, you should be familiar with the following:

- How BRM performs authentication, authorization, and accounting (AAA) for prepaid services. See "How BRM Processes Prepaid AAA Requests".
- How lightweight authorization allows you to improve performance of prepaid AAA by bypassing the rating and discounting engines for authorization in certain cases.

About Lightweight Authorization

Lightweight authorization allows you to improve performance of prepaid AAA by bypassing the rating and discounting engines for authorization in certain cases. Lightweight authorization improves the performance of authorization and reauthorization requests by doing the following:

- Reducing authorization latencies.

  To reduce authorization latencies, you can authorize customers who have a prepaid balance that is above a threshold limit to access services without requiring BRM to make calls to the rating and discounting engines. You can also reauthorize customers without going through the rating process as long as their prepaid balance is above the threshold limit. You can also reject authorization and reauthorization requests without calling the rating engine when customers have no resources.

  See "About Reducing Authorization Latencies".

- Reducing network spikes during a tariff change.

  Network spikes occur when a tariff change triggers reauthorizations for customer accounts. BRM assigns a validity period when a customer logs in that expires when the rates change. At the end of the validity period, BRM forces a reauthorization with the new rates. Because these reauthorization requests occur at the tariff change, this causes a spike in the network traffic.

  The reauthorizations make sure customers have enough resources to cover the new peak or non-peak rates for the service they are using. To reduce network spikes during a tariff change, you can reauthorize customers who have more resources in their accounts at a later time than those with fewer resources. Because

the delayed reauthorization time is based on the customer's resource balance, the reauthorization requests are more evenly distributed.

See "About Reducing Network Spikes during a Tariff Change".

You can configure lightweight authorization differently for each service. For example, you might configure lightweight authorization for both authorizations and reauthorizations for the service **/service/telco/gprs** but configure it for only authorizations for the service **/service/telco/gsm/telephony**. The data you configure for each service type is stored in the **/config/auth_reauth_info** object in the BRM database.

For information on configuring lightweight authorization, see "Setting Up Lightweight Authorization".

Note: Lightweight authorization supports branding.

About Traffic-Light Status

Lightweight authorization uses a green-, yellow-, and red-light status for requiring rating for authorization and reauthorization requests. Table 8–1 summarizes how BRM handles authorization or reauthorization requests for each traffic light status:

Table 8–1 BRM Actions and Traffic-Light Status

Traffic-Light Status	BRM Action
Green	BRM approves the authorization request without making calls to the rating and discounting engines. **Note:** For a green-light status, BRM can deduct a reservation amount from the customer's resource balance to act as a security deposit. See "About the Reservation Amount".
Yellow	BRM calls the rating and discounting engines in calc-only mode to determine if the authorization request should be approved or rejected.
Red	BRM rejects the authorization request without making calls to the rating and discounting engines.

BRM uses a threshold value called the upper threshold for each resource in a given service to determine if the request has a green or yellow status. BRM uses a credit limit for each resource in a given service to determine if the request has a yellow or red status.

- The *upper threshold* is the minimum account balance required for an authorization request to be automatically approved. See "About the Upper Threshold".
- The *credit limit* is the minimum account balance required to use a service. Credit limits are typically set to zero. When an account balance does not meet the minimum credit limit, BRM automatically rejects the authorization request.

How BRM Determines the Traffic-Light Status

Because prepaid balances are represented as a negative value, prepaid balances start at a negative number and go towards zero as the customer uses those resources. For example, when you credit a customer's account with $25, the prepaid balance is -25; when the customer spends $10, the prepaid balance changes to -15.

Upper threshold and credit limit values are set to negative numbers as well. For example, to have an upper threshold of $25, you set the upper threshold to -25. Likewise, to have a credit limit of $2, you set the credit limit to -2.

Table 8–2 summarizes how BRM sets the traffic-light status:

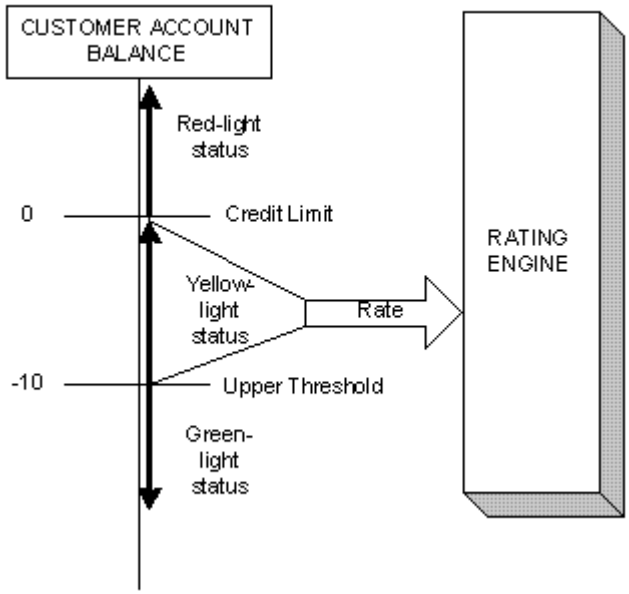
Table 8–2 How BRM Sets Traffic-Light Status

Traffic-Light Status	Description
Green	The customer's current resource balance is less than the upper threshold value you set for that resource. For example, the upper threshold is -10 and the current balance is -25. **balance < upper threshold**
Yellow	The customer's current resource balance is the same as or greater than the upper threshold value you set for that resource, but the resource balance has not exceeded its credit limit. For example, the upper threshold is -10, the current balance is -5, and the credit limit is 0. **credit limit > balance => Upper threshold**
Red	The customer's resource balance is equal to or greater than the credit limit for that service. For example, the current balance is 0 and the credit limit is 0. **balance => credit limit**

For example, if you set the upper threshold to -10 and the credit limit to 0 as shown in Figure 8–1, BRM handles authorizations as follows:

- A customer who has a resource balance of -13 is authorized with a green-light status. BRM approves the authorization without calling the rating and discounting engines.
- A customer who has a resource balance of -8 is authorized with a yellow-light status. BRM calls the rating and discounting engines.
- A customer who has a resource balance of 0 is denied authorization. BRM rejects the authorization without going through the rating process.

Figure 8–1 Traffic-Light Status



About Traffic-Light Status and Multiple Resources in a Service

If there are multiple resources for the given service, BRM checks the account balance for all resources for the service and compares the balance of each resource with the upper threshold value set for each corresponding resource. BRM applies the following rules for traffic light status when there are multiple resources:

- If any resources are green = green-light status
- If any resources are yellow and none are green = yellow-light status
- If all resources are red = red-light status

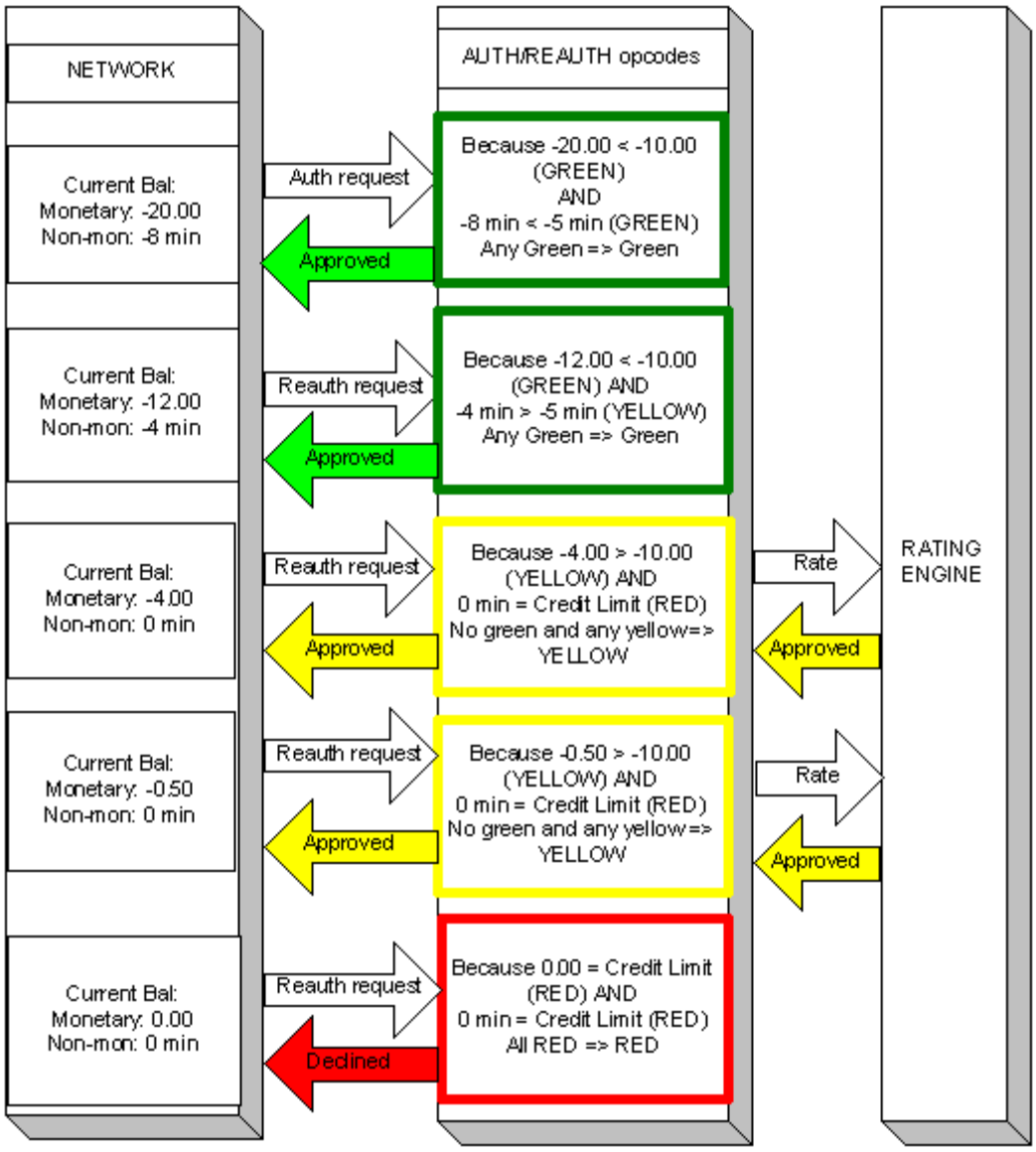
For example, if two resources have a red-light status (reached their credit limits) and one resource has a green-light status, the request is authorized with a green-light status.

Figure 8–2 illustrates how BRM handles authorization and reauthorization requests when lightweight authorization is configured for the **/service/telco/gsm/telephony** service that offers both dollar and minute resources. Prepaid balances are represented as negative values, where the upper threshold for dollars is -10 and the upper threshold for minutes is -5 (Table 8–3):

Table 8–3 Upper Thresholds for Resources

Resource	Upper Threshold
Dollars	-10
Minutes	-5

Figure 8–2 Lightweight Authorization Example



About Traffic-Light Status and Sponsored Accounts

If you offer sponsored accounts, a sponsor can have sufficient resources to cover the cost of a call when the sponsoree does not have resources. To avoid rejecting the authorization request of the sponsoree in this case, set the **AllowQuickReject** entry to **0**. By setting this entry to **0**, BRM uses a yellow-light status rather than a red-light status for a sponsoree that has no resources. See "Configuring Lightweight Authorization".

About Reducing Authorization Latencies

BRM puts every authorization and reauthorization request through the calc-only rating process by making calls to the rating and discounting engines to accurately determine if a customer call can be approved or rejected based on rating results. You can configure BRM to approve authorization or reauthorization requests without going through the rating process when prepaid customers have sufficient funds or

resources for a given service. You can also reject customer authorization or reauthorization requests without going through the rating process when they have no resources. By avoiding the rating process, you increase the number of authorization and reauthorization requests BRM can process per second for prepaid accounts.

The traffic-light status of the authorization request is what determines whether the rating and discounting engines will be called. See "About Traffic-Light Status".

To configure BRM to reduce authorization latencies, see "Setting Up Lightweight Authorization".

About the Upper Threshold

The upper threshold is the amount of resource, set for each resource in a service, that determines if an authorization request has a green- or yellow-light traffic-light status.

Green-light authorizations pose the risk that customers will exceed their credit limits before reauthorizations are sent because a validity period has not been obtained from the rating engine. For customers who are on line for the same amount of time, a lower value for the upper threshold poses less risk of revenue loss because it provides more time in which a reauthorization can occur after the customer's resource balance crosses into a yellow-light status.

Example: An upper threshold of $10 is more conservative than an upper threshold of $5.

Fred and Wilma each intend to make a 10-minute call at a rate of $1 per minute.

Fred has an account balance of $12 when he is authorized. The upper threshold is $10, so he gets a green-light status. After 8 minutes, Fred is reauthorized. He now has an account balance of $4. He gets a yellow-light status, which calls the rating engine and sets his validity period to 4 minutes. He hangs up after 2 minutes, leaving $2 in his account. If he had continued to talk, his call would have been dropped after 2 minutes because his validity period was known.

Wilma has an account balance of $6. The upper threshold is $5, so she gets a green-light status. After 8 minutes, Wilma is reauthorized. She has exceeded her credit limit by $2. Her call is dropped after a revenue loss of $2. Because the upper threshold was set so close to the credit limit, the reauthorization did not occur in time to prevent revenue loss.

To specify the upper threshold, set the **UpperThreshold** entry in the **pin_config_auth_reauth_info.xml** file. See "Configuring Lightweight Authorization".

For more information on traffic-light status, see "About Traffic-Light Status".

About the Reservation Amount

Important: Using a reservation amount decreases performance.

The reservation amount is the "security deposit" you can deduct from a customer's resource balance during green-light authorization and each green-light reauthorization. The amount represents whatever resource you are setting the reservation for, such as minutes, dollars, or frequent flyer miles.

Note: If you do *not* plan to use lightweight authorization for *re*authorizations, you do not need to set a reservation amount.

This reservation is used specifically for lightweight authorization because the rating and discounting engines are not called for green-light authorizations. BRM can also set aside a reservation amount to prevent customers from applying resources to other services while a session is in progress through concurrent network sessions (see "About Reserving Resources for Prepaid Services"). If other reservations exist, this reservation adds on to them. BRM uses the amount you specify to extend the reservation amount of the **/reservation** object.

BRM releases this reservation when the customer reaches a yellow-light status (when the rating engine is called). When the reservation amount is released, unused resources are returned to the customer's prepaid account balance.

To specify the reservation amount, set the **ReservedAmt** entry in the **pin_config_auth_reauth_info.xml** file. See "Configuring Lightweight Authorization".

For more information on traffic-light status, see "About Traffic-Light Status".

About Enabling Lightweight Authorization for Reauthorization Requests

You can use lightweight authorization when you only authorize requests (default) or when you both authorize and reauthorize requests.

To enable lightweight authorization for reauthorization requests, set the **ReauthFlag** entry in the **pin_config_auth_reauth_info.xml** file to **1**. See "Configuring Lightweight Authorization".

When this entry is set to **0** for the specified service, BRM sends a yellow-light status so that normal rating occurs for the reauthorization of that service.

> **Note:** BRM sends the scaled delay time for reducing network spikes during a tariff change even when the **ReauthFlag** entry is set to **0**. Thus, you can use that feature even if you choose to disable lightweight authorization for reauthorization requests. See "About Reducing Network Spikes during a Tariff Change".

How BRM Reduces Authorization Latencies

BRM reduces authorization latencies for prepaid accounts by eliminating calls to the rating and discounting engines for authorizations and reauthorizations that have a green-light or red-light traffic-light status.

The PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD opcode, the policy opcode PCM_OP_ACT_POL_SET_RESOURCE_STATUS, and the BRM authorization and reauthorization opcodes work together to reduce prepaid authorization latencies as follows:

1. PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD takes in the account and its related services and reads the **/config/auth_reauth_info** object to obtain lightweight authorization threshold data for each service.
2. PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD does the following for each service:

   If a service is not found in the **/config/auth_reauth_info** object, sets its status to a yellow light.

   If a service is found in the **/config/auth_reauth_info** object, sets the prepaid-traffic light status as follows:

- Sets a green-light status if the current resource balance is less than the upper threshold (for example, the resource balance is $20 and the upper threshold is $10).
- Sets a red-light status if all resource balances are above (have exceeded) the credit limit, unless the PIN_FLD_ALLOW_QUICK_REJECT field has been set to **0** for the service (which sets a yellow-light status instead, see "About Traffic-Light Status and Sponsored Accounts").
- Sets a yellow-light status for reauthorization requests when the REAUTH_FLAG flag is set to **0** (see "About Enabling Lightweight Authorization for Reauthorization Requests").

3. PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD calls the policy opcode PCM_OP_ACT_POL_SET_RESOURCE_STATUS. You can override the traffic-light status, reservation amount, and scaled delay time for the service of a particular account based on your custom requirements.

   Note: By default, the new policy opcode returns the same traffic-light status, reservation amount, and scaled delay time as calculated by the PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD opcode. However, depending on your business requirements and custom rules, this opcode can override the traffic-light status, reservation amount, and scaled delay time for the service of a particular account.

4. The following authorization and reauthorization opcodes call PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD to obtain the prepaid traffic-light status of the service:
   - PCM_OP_TCF_AAA_AUTHORIZE
   - PCM_OP_TCF_AAA_REAUTHORIZE
   - PCM_OP_ACT_AUTHORIZE
   - PCM_OP_ACT_REAUTHORIZE

   If an authorization request has a green-light status with no reservation amount or a red-light status, the PCM_OP_TCF_AAA_AUTHORIZE (for authorizations) and PCM_OP_TCF_AAA_REAUTHORIZE (for reauthorizations) opcodes immediately approve or reject the request accordingly without making calls to the rating and discounting engines.

   If an authorization request has a green-light status *with* a reservation amount, the PCM_OP_ACT_AUTHORIZE (for authorizations) and PCM_OP_ACT_REAUTHORIZE (for reauthorizations) opcodes approve the request immediately without making calls to the rating and discounting engines.

For detailed information on how BRM authorizes and reauthorizes users when lightweight authorization is configured, see the following:

- How BRM Authorizes Users to Access Services When Lightweight Authorization Is Configured
- How BRM Reauthorizes Prepaid Services When Lightweight Authorization Is Configured.

About Reducing Network Spikes during a Tariff Change

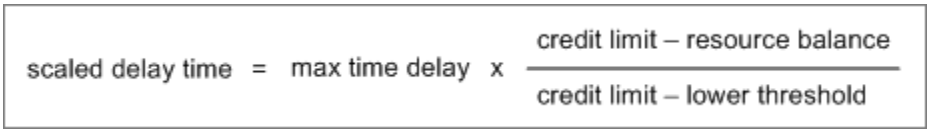
During a tariff change, network elements, such as Services Framework, request a reauthorization for non-duration-based services (volume-based services) to verify that the service contains sufficient resources to cover the new rates. BRM assigns a validity period when a customer logs in that expires when the rates change. At the end of the validity period, BRM forces a reauthorization with the new rates. Because these reauthorization requests occur at the tariff change, this causes a spike in the network traffic.

To evenly distribute these reauthorization requests, BRM calculates a scaled delay time based on the available resources in the given service, which you can use to delay the reauthorization.

BRM calculates the scaled delay time according to the values you configure for the maximum time delay and the lower threshold. For information on how the maximum time delay and the lower threshold affect the scaled delay time, see "About the Lower Threshold" and "About the Maximum Time Delay".

The scaled delay time is calculated as shown in Figure 8–3 by multiplying the maximum time delay by a scaling factor. The scaling factor is the credit limit (which is typically 0) minus the customer's resource balance divided by the credit limit minus the lower threshold:

Figure 8–3 Scaled Delay Time Calculation



To configure BRM to reduce network spikes during a tariff change, see "Setting Up Lightweight Authorization".

About the Lower Threshold

BRM uses the lower threshold in conjunction with the maximum time delay to calculate the scaled delay time. The value represents the resource for which it is configured.

BRM recommends you set the lower threshold to a lower value than the upper threshold (the entry that sets the threshold for a green-light status). For example, set an upper threshold of $10 and a lower threshold of $30.

For a given account balance and a given maximum time delay, the lower the value for the lower threshold, the shorter the scaled delay time and the earlier the reauthorization.

For example, if the maximum time delay is 20 minutes, a prepaid account authorized with a Dollar resource balance of $20 has a scaled delay time of 16 minutes if the lower threshold is $25 or 8 minutes if the lower threshold is $50.

About the Maximum Time Delay

The maximum time delay is the maximum delay that can be returned to the caller. The value is passed in as a number. You specify whether the value is in seconds or minutes in your custom code.

The higher the value of the maximum time delay, the longer the scaled delay time and, thus, the later the reauthorization. If you have a large customer base, you may need to

increase the maximum time delay to spread out the reauthorizations over a longer period of time to reduce network spikes more effectively.

How BRM Uses a Scaled Delay Time to Reduce Network Spikes during a Tariff Change

BRM calculates a scaled delay time you can use to more evenly distribute reauthorizations during a tariff change as follows:

1. The PCM_OP_ACT_AUTHORIZE or PCM_OP_ACT_REAUTHORIZE opcode calls the PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD opcode to obtain the service's traffic-light status and the scaled delay time.
2. PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD calculates the scaled delay time (PIN_FLD_SCALED_DELAY_TIME). See "About Reducing Network Spikes during a Tariff Change".
3. PCM_OP_ACT_AUTHORIZE or PCM_OP_ACT_REAUTHORIZE passes the traffic-light status and the scaled delay time to the PCM_OP_ACT_POL_POST_AUTHORIZE and PCM_OP_ACT_POL_POST_REAUTHORIZE policy opcodes.

   Note: You can modify these policy opcodes to customize the scaled delay time if desired. See "Customizing the Scaled Delay Time".

4. PCM_OP_ACT_AUTHORIZE or PCM_OP_ACT_REAUTHORIZE passes the scaled delay time to the Services Framework opcodes.

   Note: The validity period (PIN_FLD_VALID_TO) is not modified. In the case of a yellow-light status when the rating engine is called, PCM_OP_ACT_AUTHORIZE and PCM_OP_ACT_REAUTHORIZE pass the validity period in addition to the scaled delay time to the Services Framework opcodes.

5. The Services Framework opcodes use the scaled delay time according to your custom code to calculate the delayed reauthorization time.

   For example, in the case of a yellow-light status, you might choose to add the scaled delay time to the validity period returned from the rating engine. In the case of a green-light status, when the rating engine is not called and the validity period is not returned, you might choose to add the scaled delay time to the current GMT time.

To configure BRM to reduce network spikes during a tariff change, see "Setting Up Lightweight Authorization".

How BRM Authorizes Users to Access Services When Lightweight Authorization Is Configured

The main opcode for authorizing prepaid services is PCM_OP_TCF_AAA_AUTHORIZE.

BRM authorizes prepaid services as follows when lightweight authorization is configured:

1. PCM_OP_TCF_AAA_AUTHORIZE calls the PCM_OP_ACT_FIND opcode to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.
2. PCM_OP_TCF_AAA_AUTHORIZE calls PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD to obtain the traffic-light status of the service. For information on how the traffic-light status is set, see "How BRM Reduces Authorization Latencies".
3. PCM_OP_TCF_AAA_AUTHORIZE does the following:
   - For services with a green-light status that have *no* reservation amount, immediately approves the authorization request without making calls to the rating and discounting engines.
   - For services with a green-light status *with* a reservation amount, continues with step 4.
   - For services with a yellow-light status, continues with step 4.
   - For services with a red-light status, immediately rejects the authorization request without making calls to the rating and discounting engines.

   Note: If a sponsoree account has no resources (a red-light status), PCM_OP_TCF_AAA_AUTHORIZE does not reject the authorization if you set the **AllowQuickReject** entry to **0** for that service. See "About Traffic-Light Status and Sponsored Accounts".
4. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
5. PCM_OP_TCF_AAA_AUTHORIZE uses the template to search for duplicate sessions. If the opcode finds a session with the same active session ID, authorization fails.
6. At the PREP_INPUT stage, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object to aggregate service-specific data. The helper opcode returns a service-specific input flist.
7. At the VALIDATE_LIFECYCLE stage, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object to validate the request if the service uses a custom life cycle. If validation succeeds, the authorization process continues. If validation fails, the request is denied.
8. PCM_OP_TCF_AAA_AUTHORIZE passes the service-specific input flist to PCM_OP_ACT_AUTHORIZE.
9. PCM_OP_ACT_AUTHORIZE calls PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD to obtain the traffic-light status of the service.
10. PCM_OP_ACT_AUTHORIZE does the following:
    - For services with a green-light status *with* a reservation amount, reserves the reservation amount and approves the authorization request without making calls to the rating and discounting engines.

      To reserve the reservation amount, calls the PCM_OP_RESERVE_CREATE opcode with an amount-based request to create the reservation amount specific to lightweight authorization in the **/reservation** object. See "About the Reservation Amount".

- For services with a yellow-light status, makes calls to the rating and discounting engines to see whether the authorization request should be approved or rejected (calls the PCM_OP_ACT_POL_PRE_AUTHORIZE policy opcode).

11. PCM_OP_ACT_AUTHORIZE calls the PCM_OP_ACT_POL_POST_AUTHORIZE policy opcode to make any specified customizations to its output flist before returning the flist to the calling opcode. By default, the policy opcode drops the PIN_FLD_RESULTS field from the flist.

For information on how BRM authorizes prepaid services when lightweight authorization is *not* configured, see "How BRM Authorizes Users to Access Prepaid Services".

For more information about prepaid authorization, see "About Authorizing Prepaid Usage".

How BRM Reauthorizes Prepaid Services When Lightweight Authorization Is Configured

The main opcode for reauthorizing prepaid services is PCM_OP_TCF_AAA_REAUTHORIZE.

BRM reauthorizes prepaid sessions as follows when lightweight authorization is configured:

1. PCM_OP_TCF_AAA_REAUTHORIZE calls PCM_OP_ACT_FIND to locate the customer's account information. The opcode returns the customer's **/account** and **/service** objects.
2. PCM_OP_TCF_AAA_REAUTHORIZE calls PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD to obtain the traffic-light status of the service. For information on how the traffic-light status is set, see "How BRM Reduces Authorization Latencies".
3. PCM_OP_TCF_AAA_REAUTHORIZE does the following:
   - For services with a green-light status that have *no* reservation amount, immediately approves the reauthorization request without making calls to the rating and discounting engines.
   - For services with a green-light status *with* a reservation amount, continues with step 4.
   - For services with a yellow-light status, continues with step 4.
   - For services with a red-light status, immediately rejects the reauthorization request without making calls to the rating and discounting engines.

     Note: If a sponsoree account has no resources (a red-light status), PCM_OP_TCF_AAA_AUTHORIZE does not reject the authorization if you set the **Allow Quick Reject** entry to **0** for that service. For more information, see "About Traffic-Light Status and Sponsored Accounts".

4. At the SEARCH_SESSION stage, PCM_OP_TCF_AAA_REAUTHORIZE calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding session objects.
5. PCM_OP_TCF_AAA_REAUTHORIZE uses the template to search for the **/active_session** object.

- If the object *is* found, the opcode calls the appropriate PREP_INPUT helper opcode to prepare a service-specific input flist and then passes the input flist to PCM_OP_ACT_REAUTHORIZE to reauthorize the call.
- If the object *is not* found, the opcode calls the appropriate PREP_INPUT helper opcode to prepare a service-specific input flist and then passes the input flist to PCM_OP_ACT_AUTHORIZE to authorize the session with the given information.

6. PCM_OP_ACT_REAUTHORIZE determines whether the session's **/active_session** object (PIN_FLD_POID in the input flist) and **/reservation_active** object (PIN_FLD_RESERVATION_OBJ in the input flist) exist.
   - If the objects exist, the opcode continues with step 7.
   - If the objects do not exist, the opcode calls PCM_OP_ACT_AUTHORIZE to perform a session authorization.

   Note: If IMDB Cache shuts down after a prepaid session begins, the session's **/active_session** and **/reservation_active** objects will no longer exist.

7. PCM_OP_ACT_REAUTHORIZE calls PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD to obtain the traffic-light status of the service.
8. PCM_OP_ACT_REAUTHORIZE does the following:
   - For services with a green-light status *with* a reservation amount, reserves the reservation amount and approves the reauthorization request without making calls to the rating and discounting engines.

     To reserve the reservation amount, calls the PCM_OP_RESERVE_EXTEND opcode with an amount-based request to extend the reservation amount specific to lightweight authorization in the **/reservation** object. For more information, see "About the Reservation Amount".

     PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_RESERVE_EXTEND opcode with the PIN_RESERVE_INCREMENTAL_AMOUNT mode.
   - For services with a yellow-light status, makes calls to the rating and discounting engines to see whether the reauthorization request should be approved or rejected (calls the PCM_OP_ACT_POL_PRE_REAUTHORIZE policy opcode).
9. PCM_OP_ACT_REAUTHORIZE calls the PCM_OP_ACT_POL_POST_REAUTHORIZE policy opcode to make any specified customizations to its output flist before returning the flist to the calling opcode. By default, the policy opcode drops the PIN_FLD_RESULTS field from the flist.

   Note: If a reauthorization request has a green-light status with no reservation amount or a red-light status, PCM_OP_TCF_AAA_REAUTHORIZE approves or rejects the request respectively without calling PCM_OP_ACT_REAUTHORIZE.

   If a reauthorization request has a green-light status with a reservation amount or a yellow-light status, PCM_OP_TCF_AAA_REAUTHORIZE calls PCM_OP_ACT_REAUTHORIZE to handle the request.

For information on how BRM reauthorizes prepaid services when lightweight authorization is *not* configured, see "How BRM Reauthorizes Prepaid Services".

For more information about prepaid reauthorization, see "About Reauthorizing Prepaid Usage".

About Setting the Reauthorization Amount

Reauthorizations are supported only in the cumulative mode (that is, the reauthorization amount or quantity represents the original authorization amount plus a requested extension amount). For example, if you authorize a customer for 10 minutes and want to reauthorize for another 10 minutes, the time sent must be 20 minutes.

You must do the following when using lightweight authorization:

- Pass the full reauthorization amount or quantity to PCM_OP_TCF_AAA_REAUTHORIZE.
- Set the PCM_OP_TCF_AAA_REAUTHORIZE opcode's PIN_FLD_AGGREGATE_MODE input flist field to **4** to specify that the reauthorization amount or quantity is cumulative.

Setting Up Lightweight Authorization

To configure lightweight authorization data, edit the **pin_config_auth_reauth_info.xml** file and load the contents of the file into the BRM database by using the **load_pin_config_auth_reauth_info** utility. The data is stored in the **/config/auth_reauth_info** object.

By default, the **/config/auth_reauth_info** object contains lightweight authorization data for default service types and resources only. You can configure data for additional services and resources in the **/config/auth_reauth_info** object during customization, as well as modify the data for the default service types and resources.

Note: This procedure configures both of these lightweight authorization features:

- Reducing authorization latencies. See "About Reducing Authorization Latencies".
- Reducing network spikes during a tariff change. See "About Reducing Network Spikes during a Tariff Change".

To configure lightweight authorization, do the following:

1. Enable lightweight authorization in the **business_params** object. See "Enabling Lightweight Authorization in BRM".
2. Configure lightweight authorization. See "Configuring Lightweight Authorization".
3. Edit the prepaid traffic-light configuration file. See "Editing the Lightweight Authorization Configuration File".

Enabling Lightweight Authorization in BRM

To enable lightweight authorization, update the **activity** parameter instance in the **/config/business_params** object by using the **pin_bus_params** utility.

1. Create an editable XML file for the **activity** parameter instance by using the following command:

   ```
   pin_bus_params -r BusParamsActivity bus_params_act.xml
   ```

   This command creates the XML file named **bus_params_act.xml.out** in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for the following line:

   ```
   <LightWeightAuthorization>disabled</LightWeightAuthorization>
   ```

3. Changed **disabled** (default) to **enabled**.

   Caution: BRM uses the XML in this file to overwrite the existing **activity** parameter 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 **activity** parameter configuration.

4. Save the file and change the file name from **bus_params_act.xml.out** to **bus_params_act.xml**.

5. Use the following command to load the change into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_act.xml
   ```

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

6. Read the object with the **testnap** utility or Object Browser to verify that all fields are correct. See "Reading an Object and Fields" in *BRM Developer's Guide*.

7. Stop and restart the Connection Manager (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*.

Configuring Lightweight Authorization

To configure authorization and reauthorization threshold data for lightweight authorization (the **/config/auth_reauth_info** object), edit the lightweight authorization configuration file (**pin_config_auth_reauth_info.xml**) and then load its contents into the BRM database:

1. Open the **pin_config_auth_reauth_info.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 Lightweight Authorization Configuration File".

3. Save the edited file.

4. Use this command to load the **pin_config_auth_reauth_info.xml** file by running the **load_pin_config_auth_reauth_info** utility from the directory in which the **pin_config_auth_reauth_info.xml** file is located:

   ```
   load_pin_config_auth_reauth_info pin_config_auth_reauth_info.xml
   ```

   If you do not run the utility from the directory in which the file is located, you must include the complete path to the file. For example:

   ```
   load_pin_config_auth_reauth_info BRM_Home/sys/data/config/pin_config_auth_
   reauth_info.xml
   ```

   Caution: The **load_pin_config_auth_reauth_info** utility overwrites existing authorization and reauthorization threshold data for lightweight authorization. If you are updating data, you cannot load new data only. You must load complete sets of data each time you run the **load_pin_config_auth_reauth_info** utility.

   Important:

   - The BRM database must be up and running.
   - To connect to the BRM database, the **load_pin_config_auth_reauth_info** utility needs a configuration (pin.conf) 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*.

   Note:

   - You can run this utility to configure lightweight authorization for different brands. The utility loads the **/config/auth_reauth_info** object to the current brand in which it logs in. For information on running utilities with a branded database, see "Configuring a Branded Database" in *BRM Managing Customers*.
   - If you copy the **pin_config_auth_reauth_info.xml** file to the directory from which you run the **load_pin_config_auth_reauth_info** utility, you do not have to specify the path or file name. By default, the file is named **pin_config_auth_reauth_info.xml**. You can change this name.

   For more information, see "Services Framework AAA Utilities".

5. Restart the CM.
6. To verify that the authorization and reauthorization threshold data was loaded, you can display the **/config/auth_reauth_info** object by using the Object Browser application in Developer Center, or using the **robj** command with the **testnap** utility.

Editing the Lightweight Authorization Configuration File

You configure the authorization and reauthorization threshold data (**/config/auth_reauth_info** object) for lightweight authorization in your BRM system in the *BRM_Home*/**sys/data/config/pin_config_auth_reauth_info.xml** file.

You edit this prepaid traffic-light configuration file to configure both of these prepaid-traffic light authorization features:

- Reducing authorization latencies. See "Editing the File to Reduce Authorization Latencies".
- Reducing network spikes during a tariff change. See "Editing the File to Reduce Network Spikes during a Tariff Change".

Editing the File to Reduce Authorization Latencies

To reduce authorization latencies, perform these tasks:

1. Open the **pin_config_auth_reauth_info.xml** file in an XML editor or text editor.
2. Set the following entries for each service that uses lightweight authorization:
   - Set the **UpperThreshold** entry for each resource in the service. See "About Reducing Authorization Latencies".
   - Set the **ReservedAmt** entry for each resource in the service. See "About the Reservation Amount".
   - Set the **ReauthFlag** entry. See "About Enabling Lightweight Authorization for Reauthorization Requests".
   - Set the **AllowQuickReject** entry. See "About Traffic-Light Status and Sponsored Accounts".

Editing the File to Reduce Network Spikes during a Tariff Change

To reduce network spikes during tariff changes, perform these tasks:

1. Open the **pin_config_auth_reauth_info.xml** file in an XML editor or text editor.
2. Set the following entries for each service that uses lightweight authorization:
   - Edit the **LowerThreshold** entry for each resource in the service. See "About the Lower Threshold" and "How BRM Uses a Scaled Delay Time to Reduce Network Spikes during a Tariff Change".
   - Edit the **MaxTimeDelay** entry. See "About the Maximum Time Delay" and "How BRM Uses a Scaled Delay Time to Reduce Network Spikes during a Tariff Change".
   - (Optional) Customize the calculation of the scaled delay time. See "Customizing the Scaled Delay Time".

Format of the Prepaid Traffic-Light Configuration File

The format of the **pin_config_auth_reauth_info.xml** file is as follows:

```
<AuthReauthInfoConfiguration>
   <ServiceConfig>
      <ServiceType>service_type_name</ServiceType>
         <ReauthFlag>reauthorization_flag</ReauthFlag>
         <MaxTimeDelay>max_time_delay</MaxTimeDelay>
          <ResourceType>
            <ResourceID>resource_id</ResourceID>
```

```
                <OnCondition>
                   <UpperThreshold>threshold_upper</UpperThreshold>
                   <LowerThreshold>threshold_lower</LowerThreshold>
                   <ReservedAmt>reservation_amt</ReservedAmt>
                </OnCondition>
          </ResourceType>
            <!-Can repeat other ResourceType>
      …
   </ServiceConfig>
   …
   <!-Can repeat other ServiceConfigs>
</AuthReauthInfoConfiguration>
```

Sample Prepaid Traffic-Light Configuration File

The following sample **pin_config_auth_reauth_info.xml** file shows a different prepaid traffic-light configuration for two services:

```
<AuthReauthInfoConfiguration
         xmlns="http://www.portal.com/schemas/BusinessConfig"
         xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig
pin_config_auth_reauth_info.xsd">

   <ServiceConfig>
      <ServiceType>/service/gsm/telephony</ServiceType>
      <ReauthFlag>1</ReauthFlag>
      <MaxTimeDelay>600</MaxTimeDelay>
      <AllowQuickReject>1</AllowQuickReject>
      <ResourceConfig ResourceId="978">
         <UpperThreshold>-10</UpperThreshold>
         <LowerThreshold>-5</LowerThreshold>
         <ReservedAmt>3</ReservedAmt>
      </ResourceConfig>
      <ResourceConfig ResourceId="250">
         <UpperThreshold>-60</UpperThreshold>
         <LowerThreshold>-30</LowerThreshold>
         <ReservedAmt>18</ReservedAmt>
      </ResourceConfig>
   </ServiceConfig>

   <ServiceConfig>
      <ServiceType>/service/telco/gsm</ServiceType>
      <ReauthFlag>1</ReauthFlag>
      <MaxTimeDelay>800</MaxTimeDelay>
      <AllowQuickReject>0</AllowQuickReject>
      <ResourceConfig ResourceId="978">
         <UpperThreshold>-15</UpperThreshold>
         <LowerThreshold>-7</LowerThreshold>
         <ReservedAmt>2</ReservedAmt>
      </ResourceConfig>
      <ResourceConfig ResourceId="250">
         <UpperThreshold>-90</UpperThreshold>
         <LowerThreshold>-48</LowerThreshold>
         <ReservedAmt>12</ReservedAmt>
      </ResourceConfig>
   </ServiceConfig>

</AuthReauthInfoConfiguration>
```

Table 8–4 summarizes the configuration entries in the **pin_config_auth_reauth_info.xml** file:

Table 8–4 Entries in pin_config_auth_reauth_info.xml File

XML File Entry	Description
ServiceType	The service type for which lightweight authorization needs to be enabled, in the format **/service/** *name* or **/service/***name***/***name*. Enter the name of a service defined in your BRM system. You can have a different prepaid traffic-light configuration for each service.
ReauthFlag	(Optional) Enables (**1**) or disables (**0**, the default) lightweight authorization for the reauthorization for the given service. See "About Enabling Lightweight Authorization for Reauthorization Requests".
MaxTimeDelay	Maximum delay time that can be returned to the caller. See "About the Maximum Time Delay".
AllowQuickReject	Disallows (**0**) a red-light status for sponsoree accounts and returns a yellow-light status instead. See "About Traffic-Light Status and Sponsored Accounts".
ResourceConfig	List of all resources for the given service for which you want to use lightweight authorization. **Note:** The **UpperThreshold**, **LowerThreshold**, and **ReservedAmt** entries must be configured for each resource in a service.
UpperThreshold	Upper threshold value to identify cases for green-light authorization. See "About the Upper Threshold". Set this entry for each resource in the service. The value represents whatever resource you are configuring, such as minutes, dollars, or frequent flyer miles. **Note:** This entry is not related to the **LowerThreshold** entry for identifying a green-light status.
LowerThreshold	Lower threshold value to calculate the time delay. It is mandatory if **MaxTimeDelay** is set. See "About the Lower Threshold". Set this entry for each resource in the service. The value represents whatever resource you are configuring, such as minutes, dollars, or frequent flyer miles. BRM recommends you use a lower value for the lower threshold than the upper threshold (for example, if the upper threshold is -20, the lower threshold is -30). **Note:** This entry is not related to the **UpperThreshold** entry for identifying a green-light status.
ReservedAmt	In case of green-light authorization, the amount of resource to be reserved. See "About the Reservation Amount". Set this entry for each resource in the service. **Note:** If you set **ReauthFlag** to **0** to disable lightweight authorization for the reauthorization for the given service, you do not need to set a reservation amount.

Customizing the Scaled Delay Time

To customize the calculation of the scaled delay time, modify the PCM_OP_ACT_POL_POST_AUTHORIZE policy opcode or the PCM_OP_ACT_POL_POST_REAUTHORIZE policy opcode. Use the PIN_FLD_RESOURCE_STATUS and PIN_FLD_SCALED_DELAY_TIME fields in these policy opcodes to revert the scaled delay time, to apply a new scale, or both.

PCM_OP_ACT_AUTHORIZE calls the following opcodes before returning values to the Services Framework:

- PCM_OP_ACT_PRE_AUTHORIZE
- PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD, which returns the scaled delay time
- PCM_OP_ACT_POL_POST_AUTHORIZE or PCM_OP_ACT_POL_POST_REAUTHORIZE

Use the PCM_OP_ACT_POL_POST_AUTHORIZE policy opcode or the PCM_OP_ACT_POL_POST_REAUTHORIZE policy opcode to modify the scaled delay time that is passed in from PCM_OP_ACT_CHECK_RESOURCE_THRESHOLD so that PCM_OP_ACT_AUTHORIZE will send back your version of the scaled delay to the Services Framework.

Overriding the Traffic-Light Status, Reservation Amount, and Scaled Delay Time

You can override the traffic-light status, reservation amount, and scaled delay time by using the PCM_OP_ACT_POL_SET_RESOURCE_STATUS opcode.

The PCM_OP_ACT_POL_SET_RESOURCE_STATUS opcode includes the following fields in the output:

- PIN_FLD_RESULT specifies the resource availability status.
- PIN_FLD_SCALED_DELAY_TIME specifies the delay time in seconds. The value can be any decimal number.
- PIN_FLD_BALANCES specifies the balance amount to be reserved during a quick authorization, namely, the green traffic-light status.

Note: BRM does not validate the values set for the resource availability status, reservation amount, and scaled delay time for the service of a particular account in the PCM_OP_ACT_POL_SET_RESOURCE_STATUS policy opcode. To avoid potential problems such as revenue leakage, it is important that you set the appropriate values.

Consider the following examples:

- A case where BRM returns a yellow-light status and you want to override it to green. Based on the customer's current resource balance, you can do so by setting the resource availability status to green and the balance amount to be reserved in the output flist of the PCM_OP_ACT_ POL_SET_RESOURCE_STATUS policy opcode.
- A case where BRM returns a red-light status and you want to override it to green. You can do so by setting the resource availability status to green and the balance amount to be reserved to zero in the output flist of the PCM_OP_ACT_ POL_SET_RESOURCE_STATUS policy opcode. In this case, the call will be authorized even though the customer has insufficient resource balance.

Note: Oracle does not ship the source code files for policy opcodes. You must create your own source code for the new policy opcode. For more information about customizing policy opcodes, see "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.

9

About Provisioning GSM Services

This chapter provides the following information:

- An overview of how Oracle Communications Billing and Revenue Management (BRM) provisioning works.
- Details of service, supplementary service, and service-level extended rating attributes (ERA) provisioning status.

Before reading this document, see "About Integrating Wireless Services" and "About Managing Prepaid Services and Extended Rating Attributes".

How GSM Provisioning Works

When customers purchase or update their GSM services, events occur that trigger wireless service provisioning.

1. BRM generates a service order and sends it to the Provisioning Data Manager (DM). The service order contains the information required for service provisioning.
2. The Provisioning Data Manager processes the request and converts the service order information to flist XML format. The service order is then sent to the provisioning system.

Actions that Trigger GSM Provisioning

Provisioning occurs whenever customer data on the network needs to be changed. For example:

- Activating, changing, and inactivating GSM services and supplementary services. (During service activation and inactivation, phone numbers and SIM cards can also be provisioned or unprovisioned.)
- Pre-provisioning SIM cards.
- Changing SIM cards, phone numbers, and other service attributes, such as call forwarding.

About Delayed Provisioning

GSM service provisioning is triggered according to the product or deal purchase date or end date:

- If the purchase or end date is not specified, the date is by default the current date. This means that the service is provisioned or unprovisioned as soon as the product or deal is purchased or canceled.
- If the purchase or end date is in the future, the service is provisioned at the future date. This is known as *delayed provisioning*.

You should use delayed provisioning whenever possible. For example, instead of performing a product upgrade on the same day it is requested, schedule the upgrade for the following day.

The advantage of using delayed provisioning is that provisioning is triggered for only the net difference between the existing supplementary services and the new supplementary services. For example, an account might own a product that includes these supplementary services:

- Voice mail
- Roaming

Using delayed activation, you cancel the existing product and add a product that includes these supplementary services:

- Voice mail
- Call blocking

In this case, BRM unprovisions roaming, and provisions call blocking. The customer's voice mail configuration, including any existing messages, is left unchanged.

Note: If upgrading products does not have a provisioning impact, you can change how to charge for services by using the RATEPLAN ERAs instead of by changing products. See information about adding pipeline account-level and service-level rate plan promotions in the Customer Center Help.

About GSM Service Provisioning Flags

GSM services use the standard BRM status attributes: active, inactive, and closed. The status of a customer's service is displayed in Customer Center. In addition, GSM services include provisioning flags that indicate provisioning status.

Service orders often include multiple provisioning requests. For example, a service order might include provisioning requests for the following:

- The bearer service.
- One or more supplementary services.
- A voice mailbox.

In some cases, only some of the provisioning requests in an order can be completed, while others fail. In most cases, if a service can be configured so that the customer can at least perform some activities, the service order status is successful. In that case, the remaining provisioning requests can be completed later. Your business policies or your provisioning configuration might be different. For example, you might specify which services or supplementary services must be provisioned before an account can be created.

By default, the provisioning flags for GSM services are not displayed in Customer Center (although you can customize Customer Center to display them). However, you can use the Event Browser to display the status of the service order. (See "Using Event

Browser to Determine the Provisioning Status".) In addition, you can use the provisioning flags to customize your business policies, use event notification to trigger e-mail, or write status changes to a log file.

If provisioning or unprovisioning fails, you must perform the provisioning operation manually on the network, or complete the processing in the provisioning system. You cannot resend the provisioning service order from Customer Center.

Table 9–1 lists the GSM service provisioning flags.

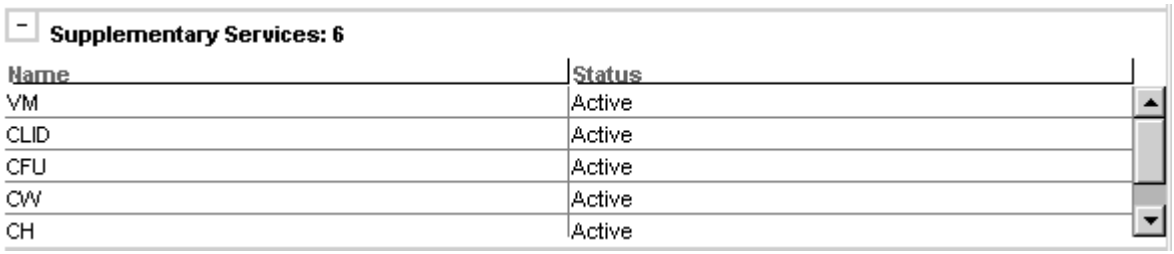
Table 9–1 Provisioning Flags for GSM Services

Provisioning Flag	Description
Processing-Provisioning	Provisioning for the GSM service is in progress. This flag is set by BRM when you create or activate a service.
Provisioning-Failed	Provisioning for the GSM service has failed. This flag is set by BRM: ▪ When provisioning fails while activating a service. ▪ When provisioning fails while inactivating or closing a service. If a service is closed but has a Provisioning-Failed flag, BRM returns an error.
Unprovisioning	The GSM service is in the process of being unprovisioned. This flag is set by BRM when you close or inactivate a service.
Suspend	The GSM service is suspended. This flag is set by BRM when you inactivate a service.

About Supplementary Service Provisioning Flags

Customer Center displays the provisioning flags of supplementary services in the **Service** tab as shown in Figure 9–1:

Figure 9–1 Supplementary Services Provisioning Flags in Customer Center



The default supplementary service provisioning flags are shown in Table 9–2:

Table 9–2 Default Supplementary Service Provisioning Flags

Provisioning Flag	Description
Provisioning	The supplementary service is in the process of being provisioned.
Provisioning Failed	Supplementary service provisioning failed.
Active	Supplementary service provisioning completed successfully.

Table 9–2 (Cont.) Default Supplementary Service Provisioning Flags

Provisioning Flag	Description
Suspending	The supplementary service is in the process of being inactivated.
Suspending Failed	Unprovisioning failed while the supplementary service was being inactivated.
Suspended	The supplementary service was successfully inactivated.
Unprovisioning	The supplementary service was canceled and the supplementary service is in the process of being unprovisioned.
Unprovisioning Failed	The supplementary service was canceled but unprovisioning failed.
Unprovisioned	The supplementary service was successfully canceled and unprovisioned.

About Service ERA Provisioning Flags

Customer Center displays the provisioning flags of service-based ERAs that have a provisioning impact. (Some ERAs do not require provisioning.) The default provisioning flags are shown in Table 9–3:

Table 9–3 Service ERA Provisioning Flags

Provisioning Flag	Description
No_op	There is no provisioning impact.
Provisioning	The ERA is in the process of being provisioned.
Provisioning Failed	ERA provisioning failed.
Active	ERA provisioning completed successfully.
Suspending	The ERA is in the process of being inactivated.
Suspending Failed	Unprovisioning failed while the ERA was being deleted.
Suspended	The ERA was successfully deleted.
Unprovisioning	The ERA was deleted and is in the process of being unprovisioned.
Unprovisioning Failed	The ERA unprovisioning failed.
Unprovisioned	The ERA was successfully unprovisioned.

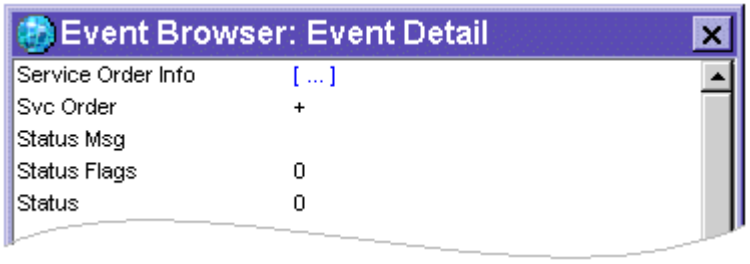
Using Event Browser to Determine the Provisioning Status

To display the provisioning status for an account:

1. Open the account in Customer Center.
2. Choose **Edit - Event Browser**.
3. Search for all events for the account.
4. In the search results, look for events of this type: **/event/provisioning/service_order/gsm**.
5. Select the events and choose **View - Event Details**.

6. To display the status of the provisioning requests in the service order, click **Service Order Info** as shown in Figure 9–2

Figure 9–2 Service Order Info Link



The information displayed shows the services and devices included in the service order as in Figure 9–3.

Figure 9–3 Services and Devices Information

POID	0.0.0.1 /service/gsm/telephony 15153 8	0.0.0.1 /device/sim 10345 1	0.0.0.1 /device/num 11075 1
Action	A	A	A
Name			
Params	[...]	[...]	[...]

7. To display the provisioning status of a service or device, click on an entry in the **Params** row.

   Figure 9–4 shows the status of a successful telephone service provisioning, including several supplementary services:

Figure 9–4 Successful Telephone Service Provisioning

Action	A	A	A	A	A	A	A	A	A
Name	CLIP	CW	BAICR	BOIC	CFU	VMBOX	ROAM	MP...	BEARER_SERVICE
Value									T11
Status	0	0	0	0	0	0	0	0	0
Status Msg									

About Customizing and Localizing GSM and Supplementary Service Provisioning Flags

You can customize and localize the provisioning flags for GSM supplementary services and service-level ERAs To do so, you edit a copy of the **features_and_profiles_states.en_US** sample file in the *BRM_Home*/**sys/msgs/featuresandprofilestates** directory. 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* and "Loading GSM Provisioning Flag Definitions".

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

```
load_localized_strings features_and_profiles_states.locale
```

Note: 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*.

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

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

About XML Provisioning

To initiate service provisioning on a carrier network, GSM Manager sends service order information to a third-party network provisioning agent in the form of XML-formatted *provisioning payload* files. The provisioning agent returns the result of the provisioning request to GSM Manager.

The basic network provisioning process:

1. Provisioning Data Manager (DM) (**dm_prov_telco**):
   a. Receives a service order.
   b. Creates an XML provisioning payload file that includes fields specified in the provisioning configuration file (**/config/provisioning/telco**).
   c. Sends it to the network provisioning agent.
2. The Provisioning DM (**dm_prov_telco**) waits for an acknowledgment from the network provisioning agent.

   This is a sample POID field from a service creation response:

   ```
   0.0.0.1 /event/provisioning/service_order/telco/gsm/telephony 18832
   ```

3. The network provisioning agent returns the provisioning result (Success or Failure) to an opcode which updates the service order using the POID field from the response.

   For more information on supported fields, see "Service Order XML DTD".

Sample XML Document

The following is a sample XML provisioning payload file generated for service creation:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
 <order>
         <POID>0.0.10.2 / 0 0</POID>
         <EVENT_OBJ>0.0.0.1 /event/provisioning/service_order/telco/gsm/telephony 18832 0</EVENT_
OBJ>
        <SVC_ORDER>
                  <STATUS>1</STATUS>
         </SVC_ORDER>
         <SERVICE_ORDER_INFO elem="0">
                  <ACTION>A</ACTION>
                  <POID>0.0.0.1 /device/sim 13947 1</POID>
                  <PARAMS elem="0">
                          <VALUE>000000020001152</VALUE>
                          <ACTION>I</ACTION>
```

```
          <NAME>IMSI</NAME>
  </PARAMS>
  <PARAMS elem="1">
          <VALUE>000000000200011526</VALUE>
          <ACTION>I</ACTION>
          <NAME>SIM</NAME>
  </PARAMS>
          </SERVICE_ORDER_INFO>
          <SERVICE_ORDER_INFO elem="1">
          <ACTION>A</ACTION>
          <POID>0.0.0.1 /device/num 10105 1</POID>
  <PARAMS elem="0">
          <VALUE>00493451212</VALUE>
          <ACTION>I</ACTION>
          <NAME>MSISDN</NAME>
  </PARAMS>
          </SERVICE_ORDER_INFO>
          <SERVICE_ORDER_INFO elem="2">
          <NAME>MOBTEL</NAME>
         <ACTION>A</ACTION>
         <POID>0.0.0.1 /service/telco/gsm/telephony 18400 8</POID>
         <PARAMS elem="0">
         <VALUE>T00</VALUE>
         <ACTION>I</ACTION>
         <NAME>BEARER_SERVICE</NAME>
</PARAMS>
<PARAMS elem="1">
         <ACTION>A</ACTION>
         <NAME>VMBOX</NAME>
</PARAMS>
<PARAMS elem="2">
         <ACTION>A</ACTION>
         <NAME>CLIP</NAME>
</PARAMS>
<PARAMS elem="3">
         <ACTION>A</ACTION>
         <NAME>CFU</NAME>
</PARAMS>
<PARAMS elem="4">
         <ACTION>A</ACTION>
         <NAME>CW</NAME>
</PARAMS>
<PARAMS elem="5">
         <ACTION>A</ACTION>
         <NAME>HOLD</NAME>
</PARAMS>
<PARAMS elem="6">
         <ACTION>A</ACTION>
         <NAME>CD</NAME>
</PARAMS>
         </SERVICE_ORDER_INFO>
         <SERVICE_ORDER_INFO elem="3">
         <ACTION>A</ACTION>
         <POID>0.0.0.1 /device/sim 13947 3</POID>
<PARAMS elem="0">
         <VALUE>000000020001152</VALUE>
         <ACTION>I</ACTION>
         <NAME>IMSI</NAME>
</PARAMS>
<PARAMS elem="1">
```

```
                        <VALUE>000000000200011526</VALUE>
                        <ACTION>I</ACTION>
                        <NAME>SIM</NAME>
                </PARAMS>
        </SERVICE_ORDER_INFO>
</order>
```

Table 9–4 describes the main fields of the payload file.

Table 9–4 Payload File Fields

Field	Description
SVC_ORDER.STATUS	The status of the service order. Possible values are (pre-append) ■ NEW ■ READY ■ PROCESSING ■ COMPLETED ■ FAILED
SVC_ORDER.STATUS_MSG	Status message for service order.
SERVICE_ORDER_INFO[0].ACTION	The action to be performed. Possible values are: ■ A (Activate) ■ D (Deactivate) ■ S (Suspend) ■ C (Change) ■ R (Reactivate) ■ I (Ignore)
SERVICE_ORDER_INFO[0].NAME	The name of the object to be provisioned. Possible values are: ■ MOBTEL ■ MOBFAX ■ MOBDATA ■ MOBSMS
SERVICE_ORDER_ INFO[0].PARAMS[*].SUB_NAME	The name of the Parameter/supplementary services/ VAS. Examples include: ■ CFU ■ CW ■ CLIP ■ VMBOX
SERVICE_ORDER_ INFO[0].PARAMS[*].SUB_VALUE	Value associated with the NAME
SERVICE_ORDER_INFO[1].*	Information about the ESN
SERVICE_ORDER_INFO[2].*	Information about the NUM

Service Order XML DTD

The following lists the XML DTD:

```
<!ELEMENT template (version?,
                    name,
                    start_line,
                    loading_controls,
                    file_type?,
                    file_format,
                    comment_line_prefix?,
                    global_record_info,
                    global_field_info,
                    records)>
          <!ELEMENT version (#PCDATA)>
          <!ELEMENT name (#PCDATA)>
          <!ELEMENT start_line (#PCDATA)>

          <!ELEMENT loading_controls ANY>
               <!ATTLIST loading_controls group_by CDATA #REQUIRED>
          <!ELEMENT file_type (#PCDATA)>
          <!ELEMENT file_format (#PCDATA)>
          <!ELEMENT comment_line_prefix ANY>

          <!ELEMENT global_record_info (record_delimiter?,
                    discard_prefix?,
                    record_type_locator?,
                    record_length_locator?,
                    ignore_record_types?,
                    record_length?)>
          <!ELEMENT record_delimiter ANY>
          <!ELEMENT discard_prefix ANY>
          <!ELEMENT record_type_locator ANY>
               <!ATTLIST record_type_locator name CDATA #IMPLIED>
               <!ATTLIST record_type_locator position CDATA #IMPLIED>
               <!ATTLIST record_type_locator start CDATA #IMPLIED>
               <!ATTLIST record_type_locator end CDATA #IMPLIED>
          <!ELEMENT record_length_locator ANY>
               <!ATTLIST record_length_locator name CDATA #IMPLIED>
               <!ATTLIST record_length_locator position CDATA #IMPLIED>
               <!ATTLIST record_length_locator start CDATA #IMPLIED>
               <!ATTLIST record_length_locator end CDATA #IMPLIED>
          <!ELEMENT ignore_record_types (ignore_record_type*)>
          <!ELEMENT ignore_record_type (#PCDATA)>
          <!ELEMENT record_length (#PCDATA)>

          <!ELEMENT global_field_info (trim_white_space?,
                    padding_char?,
                    field_delimiter?,
                    justification?,
                    attribute_value_separator?,
                    consecutive_delimiters_is_one?,
                    literal_indicator?,
                    missing_field_indicator?)>
          <!ELEMENT trim_white_space (#PCDATA)>
          <!ELEMENT padding_char (#PCDATA)>
          <!ELEMENT field_delimiter ANY>
          <!ELEMENT justification (#PCDATA)>
          <!ELEMENT attribute_value_separator ANY>
          <!ELEMENT consecutive_delimiters_is_one (#PCDATA)>
          <!ELEMENT literal_indicator ANY>
```

```
<!ELEMENT missing_field_indicator ANY>

<!ELEMENT records (record*)>
<!ELEMENT record (field_definitions?,
                  user_mapping_info?,
                  event_mapping_info?,
                  filters?,
                  checks?)>
     <!ATTLIST record name CDATA #REQUIRED>
     <!ATTLIST record type CDATA #IMPLIED>
     <!ATTLIST record record_length CDATA #IMPLIED>
     <!ATTLIST record event_class CDATA #IMPLIED>
     <!ATTLIST record service_class CDATA #IMPLIED>
     <!ATTLIST record event_opcode_name CDATA #IMPLIED>
     <!ATTLIST record event_opcode_num CDATA #IMPLIED>
     <!ATTLIST record user_info_opcode_name CDATA #IMPLIED>
     <!ATTLIST record user_info_opcode_num CDATA #IMPLIED>
<!ELEMENT field_definitions (field*)>
<!ELEMENT field (field*)>
     <!ATTLIST field name CDATA #IMPLIED>
     <!ATTLIST field start CDATA #IMPLIED>
     <!ATTLIST field end CDATA #IMPLIED>
     <!ATTLIST field position CDATA #IMPLIED>
     <!ATTLIST field data_type CDATA #IMPLIED>
     <!ATTLIST field override_value CDATA #IMPLIED>
     <!ATTLIST field default_value CDATA #IMPLIED>
     <!ATTLIST field date_format_string CDATA #IMPLIED>
     <!ATTLIST field inf_data_type CDATA #IMPLIED>
     <!ATTLIST field inf_data_type_num CDATA #IMPLIED>
     <!ATTLIST field inf_field_name CDATA #IMPLIED>
     <!ATTLIST field inf_field_num CDATA #IMPLIED>
     <!ATTLIST field elem_num CDATA #IMPLIED>
     <!ATTLIST field elem_position CDATA #IMPLIED>

<!ELEMENT user_mapping_info (field*)>
<!ELEMENT event_mapping_info (field*)>
<!ELEMENT filters (filter*)>
<!ELEMENT filter (regexp, infix_regexp?)>
     <!ATTLIST filter name CDATA #REQUIRED>
     <!ATTLIST filter discard CDATA #REQUIRED>
     <!ATTLIST filter log CDATA #REQUIRED>
<!ELEMENT regexp ANY>
<!ELEMENT infix_regexp ANY>
<!ELEMENT checks (check*)>
<!ELEMENT check (#PCDATA)>
     <!ATTLIST check name CDATA #REQUIRED>
     <!ATTLIST check type CDATA #REQUIRED>
     <!ATTLIST check expr CDATA #REQUIRED>
     <!ATTLIST check field_name CDATA #IMPLIED>
     <!ATTLIST check field_pos CDATA #REQUIRED>
```

Part III

Managing GSM Services

Part III describes how to manage GSM services in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

10

About Performing AAA for Prepaid GSM Services

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) GSM AAA Manager and describes how to implement AAA functionality in your BRM system.

Before you read this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

About Processing AAA Requests for GSM Services

GSM AAA Manager allows you to perform authentication, authorization, and accounting (AAA) for prepaid GSM services, such as GSM telephone calls, SMS messages, data, and faxes. For example, when a prepaid customer accesses a GSM service, you can use GSM AAA Manager to perform the following:

- Verify the customer's identity by using the phone's MSID.
- Determine whether the customer's account balance has enough resources to cover the cost of usage.
- Reserve a portion of the customer's resources for the GSM session.
- Record usage information about the GSM session while it is in progress.
- When the session ends, rate any usage and update the customer's account balance.

For more information about how BRM performs prepaid AAA, see "Understanding Prepaid AAA".

About GSM AAA Manager

GSM AAA Manager is an API that consists of opcodes, storable classes, and utilities that allow you to quickly implement AAA support for GSM services.

About the GSM AAA Manager Opcodes

GSM AAA Manager includes two types of opcodes:

- **Services Framework AAA standard opcodes**. Services Framework AAA standard opcodes pass AAA requests to the BRM PCM API. These opcodes are abstract opcodes for processing AAA requests for any prepaid service type. Because the Services Framework AAA opcodes are abstract, they cannot perform GSM-specific operations by themselves. They call helper opcodes to do this.

- **GSM AAA helper opcodes**. The GSM AAA helper opcodes perform GSM-specific operations, such as building search templates for GSM session objects or preparing GSM-specific flists, for the Services Framework AAA standard opcodes. See "Preparing GSM-Specific Data by Using Helper Opcodes".

A Services Framework AAA opcode calls a GSM AAA helper opcode at any of these processing stages in the opcode's execution:

- SEARCH_SESSION
- PREP_INPUT
- VALIDATE_LIFECYCLE
- TAG_SESSION
- ACC_ON_OFF_SEARCH

The Services Framework AAA opcode determines which helper opcode to call at each processing stage by reading the **/config/opcodemap/tcf** object.

For example, at the PREP_INPUT processing stage, the Services Framework AAA opcode calls the GSM AAA helper opcode specified in the **/config/opcodemap/tcf** object. The GSM AAA helper opcode aggregates the GSM data and then returns a GSM-specific flist to the Services Framework AAA opcode. The Services Framework AAA opcode passes the flist to the PCM API, which processes the request and then returns that the request either passed or failed.

For more information, see "Services Framework AAA Manager Process Overview".

By default, the Services Framework AAA opcodes call GSM AAA helper opcodes when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** services only. You can add support for additional service types or change which helper opcodes are called by using the "load_aaa_config_opcodemap_tcf" utility. See "Configuring Services Framework to Call Helper Opcodes".

About the GSM AAA Manager Storable Classes

By default, BRM stores information for prepaid GSM sessions in these storable classes:

- **/active_session/telco/gsm**: Stores information about a GSM session while it is *in progress*. This object can be subclassed for specific GSM services.
- **/session/telco/gsm**: Stores information about a GSM session that has been *rated and closed*. This object can be subclassed for specific GSM services.
- **/reservation/active**: Stores information about a single reservation for one balance group.
- **/reservation_list**: Tracks the total resources a balance group has reserved in **/reservation/active** objects.

  For policy-driven charging sessions, this object also holds the consumed reservation amount for those resources.
- **/config/aaa/gsm**: Stores default preferences for GSM services. For example, it specifies whether to keep or delete active session objects when a GSM session ends.
- **/config/reserve/gsm**: Stores the default authorization and reauthorization values for GSM services. This object can be subclassed for specific GSM services.

About the GSM AAA Manager Utilities

GSM AAA Manager utilities specify default preferences for processing GSM services, including the following:

- Whether to keep or delete **/active_session** and **/reservation/active** objects when a prepaid GSM session ends.
- Whether to check for duplicate **/active_session** or **/event/session** objects.
- The expiration time interval for **/active_session** objects stored in memory.
- Default authorization and reauthorization values for GSM services. BRM authorizes prepaid customers to use a service for a specified duration, volume, or amount. For example, BRM can authorize customers to initially make a 10-minute GSM telephone call or download 100 bytes of data.

See "Specifying Default AAA Preferences for GSM Services".

Setting Up Your System to Perform AAA for Prepaid GSM Services

To set up your system to process AAA requests for prepaid GSM services, perform the following tasks:

1. Specify your default preferences for prepaid GSM services. See "Specifying Default AAA Preferences for GSM Services".
2. Configure the gateway application to send GSM AAA requests to the GSM AAA Manager opcodes. See "Sending AAA Requests to GSM AAA Manager".
3. (Optional) Customize the GSM data used by helper opcodes. See "Preparing GSM-Specific Data by Using Helper Opcodes".
4. (Optional) Configure your system to perform AAA for custom RUMs. See "Configuring Services Framework AAA Manager for Custom RUMs".
5. (Optional) Modify which helper opcodes are called by the Services Framework AAA opcodes or the service types that are supported. See "Configuring Services Framework to Call Helper Opcodes".

   Note: By default, the Services Framework AAA opcodes call GSM AAA helper opcodes when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events only. To call the helper opcodes when processing additional GSM service types, you must configure Services Framework Manager to do so.

6. (Optional) Configure BRM to reserve a portion of a customer's resources for a prepaid GSM session by installing and configuring Resource Reservation Manager. See "Reserving Resources for Concurrent Network Sessions" in *BRM Configuring and Collecting Payments*.

Specifying Default AAA Preferences for GSM Services

You specify how BRM processes AAA requests for GSM services by using the Services Framework AAA Manager utilities and configuration files:

- To specify the default authorization and reauthorization values for GSM services, see "Specifying Default Authorization and Reauthorization Values".

- To specify how to handle GSM session objects, see "Configuring How Services Framework AAA Manages Session Objects".
- To specify the service-specific helper opcodes to call, see "Configuring Services Framework to Call Helper Opcodes".
- To specify how to calculate the total resources reserved by an account, see "Configuring How BRM Calculates Reservation Balances".

Sending AAA Requests to GSM AAA Manager

To perform AAA for GSM services, your system must be designed to collect the information needed from the customer and pass the appropriate fields in the input flist to the GSM AAA Manager opcodes.

Your external network can pass information to the GSM AAA opcodes through Oracle Communications Service Broker (OCSB) or a custom gateway application.

You can use the GSM AAA Manager opcodes to perform the following:

- Authenticate customers. See "Authenticating Users for GSM Services".
- Authorize customers to access GSM services. See "Authorizing GSM Services".
- Reauthorize customers to continue an existing GSM session. See "Reauthorizing GSM Sessions".
- Reauthorize sessions based on a customer's current usage. See "Updating and Reauthorizing GSM Sessions".
- Cancel existing authorizations. See "Canceling Authorization for GSM Services".
- Manage GSM sessions while they are in progress. See "Managing Prepaid GSM Sessions".
- Customize GSM authorization IDs. See "Customizing GSM Authorization IDs".

Authenticating Users for GSM Services

To authenticate GSM users, call the PCM_OP_TCF_AAA_AUTHENTICATE opcode with the following information in the input flist:

- GSM service type
- MSID
- Name of the calling program
- (Optional) Password (if the PIN_FLD_PASSWORD field is passed in, BRM authenticates in PAP mode; if the field is not passed in, BRM authenticates in CHAP mode)
- (Optional) Action

For more information, see "How BRM Authenticates Prepaid Customers".

Authorizing GSM Services

To authorize prepaid customers to use GSM services, call the PCM_OP_TCF_AAA_AUTHORIZE opcode with the following information in the input flist:

- GSM service type
- MSID

- Name of the calling program
- Direction of the call
- Details about the call, such as the IMEI value, the dialed number, and the quality of service (QoS)

For more information, see "How BRM Authorizes Users to Access Prepaid Services".

By default, PCM_OP_TCF_AAA_AUTHORIZE calls the helper opcodes shown in Table 10–1 when it processes **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–1 Helper Opcodes Called for Processing

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GSM_AAA_POL_AUTHORIZE_PREP_INPUT
VALIDATE_LIFECYCLE	PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE (for **/service/telco/gsm/telephony** only)

Note: To configure PCM_OP_TCF_AAA_AUTHORIZE to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Reauthorizing GSM Sessions

To reauthorize a customer to continue a GSM session, call the PCM_OP_TCF_AAA_REAUTHORIZE opcode with the following information in the input flist:

- GSM service type
- MSID
- Name of the calling program
- Direction of the call
- Requested reauthorization amount or quantity
- Details about the call, such as the IMEI value, the dialed number, and the QoS

You can specify whether the reauthorization amount or quantity is aggregated or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **4** specifies that the reauthorization amount or quantity passed in the input flist is *aggregated* (that is, it represents the original authorization amount plus a requested extension amount). BRM reauthorizes by using the amount or quantity passed in the input flist.
- **8** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the requested extension amount or quantity only). This is the default.

Note: If you specify incremental mode (**8**), you can also specify how to calculate the reauthorization amount or quantity by setting the PIN_FLD_RATING_MODE input flist field to the following:

- **0**: Calculates the reauthorization amount or quantity by adding the value from the **/reservation/active** object to the value passed in the input flist. This is the default.
- **1**: Calculates the reauthorization amount or quantity by adding the value from the **/active_session** object to the value passed in the input flist.

For more information, see "How BRM Reauthorizes Prepaid Services".

By default, PCM_OP_TCF_AAA_REAUTHORIZE calls the helper opcodes shown in Table 10–2 when it processes **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–2 Helper Opcodes Called for Reauthorization

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GSM_AAA_POL_REAUTHORIZE_PREP_INPUT
POST_PROCESS	PCM_OP_GSM_AAA_POL_POST_PROCESS

Note: To configure PCM_OP_TCF_AAA_REAUTHORIZE to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Updating and Reauthorizing GSM Sessions

To update customer usage data and reauthorize a prepaid GSM session, call the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode with the following information in the input flist:

- GSM service type
- MSID
- Name of the calling program
- Direction of the call
- Consumed quantity or amount
- Requested reauthorization quantity or amount
- Details about the call, such as the IMEI value, the dialed number, and the QoS

You can specify whether the reauthorization amount or quantity is aggregated or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the update amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session).
- **2** specifies that the update amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last

updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the value in the **/active_session** object. This is the default update mode.

- **4** specifies that the reauthorization amount or quantity passed in the input flist is *aggregated* (that is, it represents the original authorization amount plus a requested extension amount). BRM reauthorizes by using the amount or quantity passed in the input flist.
- **8** specifies that the reauthorization amount or quantity passed in the input flist is *incremental* (that is, it represents the requested extension amount or quantity only). This is the default reauthorization mode.

> **Note:** If you specify incremental mode (**8**), you can also specify how to calculate the reauthorization amount or quantity by setting the PIN_FLD_RATING_MODE input flist field to the following:
>
> - **0**: Calculates the reauthorization amount or quantity by adding the value from the **/reservation/active** object to the value passed in the input flist. This is the default.
> - **1**: Calculates the reauthorization amount or quantity by adding the value from the **/active_session** object to the value passed in the input flist.

For more information, see "How BRM Updates and Reauthorizes Prepaid Sessions".

By default, PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the following helper opcodes when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–3 Helper Opcodes Called to Update and Reauthorize

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GSM_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT and then PCM_OP_GSM_AAA_POL_REAUTHORIZE_PREP_INPUT
POST_PROCESS	PCM_OP_GSM_AAA_POL_POST_PROCESS

> **Note:** To configure PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Canceling Authorization for GSM Services

To cancel an existing authorization and return reserved resources back to the customer's account balance, call the PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION opcode with the following information in the input flist:

- GSM service type
- MSID
- Name of the calling program

- Authorization ID

For more information, see "How BRM Cancels Prepaid Service Authorizations".

By default, PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION calls the helper opcodes shown in Table 10–4 when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–4 Help Opcode Called for Cancellations

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION

Note: To configure PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Managing Prepaid GSM Sessions

After a customer is authorized to access a GSM service, the external network connects the call and begins collecting information about the customer's usage, such as the starting time, the dialed number, and the direction of the call. The network sends this information to BRM, which records the information in **/active_session** objects.

When the session ends, BRM rates any usage, closes or deletes the associated reservation and active session objects, and records the data in **/event/session** objects in the BRM database.

You use the GSM AAA ACCOUNTING standard opcodes to perform the following tasks:

- *Start* prepaid GSM sessions.
- *Update* information about a prepaid GSM session that is in progress.
- *End* prepaid GSM sessions.
- *Close* any open GSM sessions when the external network shuts down abnormally or restarts.

Starting Prepaid GSM Sessions

To start a prepaid GSM session, call the PCM_OP_TCF_AAA_START_ACCOUNTING opcode with the following information in the input flist:

- GSM service type
- MSID
- Authorization ID
- Session start time
- Direction of the call
- Name of the calling program
- Information about the call, such as the IMEI value, dialed number, and QoS

For more information, see "How BRM Starts Prepaid Sessions".

By default, PCM_OP_TCF_AAA_START_ACCOUNTING calls the helper opcode shown in Table 10–5 when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–5 Helper Opcodes Called For Start of Accounting

Processing Stage	Helper Opcode Called
SEARCH_ SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION

Note: To configure PCM_OP_TCF_AAA_START_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Updating a Prepaid GSM Session

To update information about an existing prepaid GSM session, call the PCM_OP_TCF_AAA_UPDATE_ACCOUNTING opcode with the following information in the input flist:

- GSM service type
- MSID
- Name of the calling program
- Authorization ID
- Session end time
- Direction of the call
- Details that changed

You can specify whether the usage amount or quantity is aggregated or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session).
- **2** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the session object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the session's previous usage amount or quantity.

For more information, see "How BRM Updates Prepaid Sessions".

By default, PCM_OP_TCF_AAA_UPDATE_ACCOUNTING calls the helper opcodes shown in Table 10–6 when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–6 Helper Opcodes Called to Update Accounting

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GSM_AAA_POL_UPDATE_ACCOUNTING_PREP_ INPUT

Note: To configure PCM_OP_TCF_AAA_UPDATE_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Ending Prepaid GSM Sessions

To end a prepaid GSM session when it completes successfully, call the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode with the following information in the input flist:

- GSM service type
- MSID
- Name of the calling program
- Direction of the call
- The amount or quantity consumed during the session
- Information about the call, such as the IMEI value, dialed number, and QoS

You use this opcode to perform the following operations:

- Close, cancel, or delete the active session object.
- Release or delete any associated reservation objects.
- Rate any usage.
- Record information about the GSM session in an **/event/session** object in the BRM database.

You can specify whether the amount or quantity consumed is aggregated or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session).
- **2** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the session object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the session's previous usage amount or quantity.

For more information, see "How BRM Ends Prepaid Sessions".

By default, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the helper opcodes shown in Table 10–7 when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–7 Helper Opcodes Called to Stop Accounting

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GSM_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GSM_AAA_POL_STOP_ACCOUNTING_PREP_INPUT
VALIDATE_LIFECYCLE	PCM_OP_TCF_AAA_VALIDATE_LIFECYCLE (for **/service/telco/gsm/telephony** in Direct Debit mode only)

Note: To configure PCM_OP_TCF_AAA_STOP_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Closing Prepaid GSM Sessions When the External Network Shuts Down

To close all open GSM sessions when the external network is being shut down or encounters problems, call the PCM_OP_TCF_AAA_ACCOUNTING_OFF opcode with the following information in the input flist:

- GSM service type
- Originating network ID (SCP name)
- Name of the calling program
- (Optional) Start time
- (Optional) Status
- (Optional) Termination cause

This opcode closes all sessions that match the criteria passed in the input flist.

GSM sessions with a status of STARTED or UPDATED are automatically rated before they are closed. You specify how BRM handles GSM sessions with a CREATED status by passing the optional PIN_FLD_ACC_FLAG input flist field:

- When this flag is passed, CREATED sessions are *rated* before they are closed.
- When the flag is not passed, CREATED sessions are *cancelled*.

For more information, see "How BRM Closes Prepaid Sessions When the External Network Shuts Down".

By default, PCM_OP_TCF_AAA_ACCOUNTING_OFF calls the helper opcode shown in Table 10–8 when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events:

Table 10–8 Helper Opcodes Called to Shut Down Accounting

Processing Stage	Helper Opcode Called
ACC_ON_OFF_SEARCH	PCM_OP_GSM_AAA_POL_ACC_ON_OFF_SEARCH

Note: To configure PCM_OP_TCF_AAA_ACCOUNTING_OFF to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Closing Prepaid GSM Sessions when the External Network Restarts

To close all open GSM sessions when the external network restarts, call the PCM_OP_TCF_AAA_ACCOUNTING_ON opcode with the following information in the input flist:

- GSM service type
- Originating network ID (SCP name)
- Name of the calling program

- (Optional) Start time
- (Optional) Status
- (Optional) Termination cause

This opcode closes all sessions that match the criteria passed in the input flist.

GSM sessions with a status of STARTED or UPDATED are automatically rated before they are closed. You specify how BRM handles GSM sessions with a CREATED status by passing the optional PIN_FLD_ACC_FLAG input flist field:

- When this flag is passed, CREATED sessions are *rated* before they are closed.
- When the flag is not passed, CREATED sessions are *canceled*.

For more information, see "How BRM Closes Prepaid Sessions When the External Network Restarts".

By default, PCM_OP_TCF_AAA_ACCOUNTING_ON calls the helper opcode shown in Table 10–9 when processing **/service/telco/gsm/data**, **/service/telco/gsm/fax**, **/service/telco/gsm/sms**, and **/service/telco/gsm/telephony** events.

Table 10–9 Helper Opcode Called to Restart Accounting

ProcessinG Stage	HelpeR Opcode Called
ACC_ON_OFF_SEARCH	PCM_OP_GSM_AAA_POL_ACC_ON_OFF_SEARCH

Note: To configure PCM_OP_TCF_AAA_ACCOUNTING_ON to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Customizing GSM Authorization IDs

Use the PCM_OP_GSM_AAA_POL_AUTHORIZE policy opcode to generate a unique authorization ID, if one is not passed in the input flist. By default, this opcode generates IDs that use the following format:

```
Calling_Number - Called_Number - Start_Time - Origin_Network
```

For example:

```
4085551212-9165551234-1095379771-Sample Network
```

However, you can customize this opcode to use another ID format.

This policy opcode is called by PCM_OP_GSM_AAA_AUTHORIZE during the authorization process.

Preparing GSM-Specific Data by Using Helper Opcodes

Use these GSM helper opcodes to prepare GSM-specific data for the Services Framework AAA opcodes.

Important: Do not call these opcodes directly. You configure an opcode to call helper opcodes by using the "load_aaa_config_opcodemap_tcf" utility. See "Configuring Services Framework to Call Helper Opcodes".

- To aggregate GSM data and then prepare an input flist, use PCM_OP_GSM_AAA_POL_PREP_INPUT. See "Preparing GSM-Specific Input Flists for Authorization".
- To build a search template for finding **/active_session/telco/gsm** or **/event/session/telco/gsm** objects, use PCM_OP_GSM_AAA_POL_SEARCH_SESSION. See "Building Search Templates for GSM Session Objects".
- To build search templates for finding **/active_session/telco/gsm** objects, use PCM_OP_GSM_AAA_POL_ACC_ON_OFF_SEARCH. See "Building Search Templates for GSM Active Session Objects".
- To aggregate return data, use PCM_OP_GSM_AAA_POL_POST_PROCESS. See "Aggregating Return GSM Data".

Preparing GSM-Specific Input Flists for Authorization

Use the PCM_OP_GSM_AAA_POL_AUTHORIZE_PREP_INPUT helper opcode to aggregate GSM data by duration, volume, or amount, and then prepare an input flist that can be used for AAA operations.

This opcode aggregates GSM data by the amount passed in the input flist. If an amount is not passed in the PIN_FLD_AMOUNT field, this opcode aggregates GSM data based on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies to rate the *amount*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Amount-Based Aggregation:

When aggregating the amount, the helper opcode prepares the PIN_FLD_BALANCES array in the PIN_FLD_RATING_INFO substruct, indexed by the currency type.

For Duration-Based Aggregation:

When aggregating the duration, the helper opcode performs the following:

1. Assigns a starting timestamp to the PIN_FLD_START_T field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct. The opcode uses the starting timestamp from the input flist or, if one is not passed in, from "pin_virtual_time" (see *BRM Developer's Guide*).

   Note: This is a temporary starting timestamp only and is later replaced with the actual starting timestamp by the reauthorization or stop accounting opcodes.

2. Assigns an ending timestamp to the PIN_FLD_END_T field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct:

- If PIN_FLD_END_T is supplied in the input flist, the opcode assigns the ending timestamp directly to the PIN_FLD_END_T field.
- If PIN_FLD_QUANTITY is supplied in the input flist, the opcode calculates the ending timestamp by adding the quantity passed in the input flist to the starting timestamp.
- If an ending timestamp or quantity is not passed in, the opcode retrieves the default authorization quantity from the **/config/reserve/gprs** object. The opcode calculates the ending timestamp by adding the default authorization quantity to the starting timestamp.

For Volume-Based Aggregation:

When aggregating the volume, the helper opcode adds the bytes uploaded and the bytes downloaded and assigns the value to the PIN_FLD_BYTES_UPLINK and PIN_FLD_BYTES_DOWNLINK fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT.PIN_FLD_TELCO_INFO substruct.

Preparing GSM-Specific Input Flists for Reauthorization

Use the PCM_OP_GSM_AAA_POL_REAUTHORIZE_PREP_INPUT helper opcode to aggregate GSM data and then prepare an flist for reauthorizing a prepaid GSM session. By default, this opcode is called by PCM_OP_TCF_AAA_REAUTHORIZE at the PREP_INPUT processing stage.

This opcode aggregates GSM data by the amount passed in the input flist. If an amount is not passed in the PIN_FLD_AMOUNT field, this opcode aggregates GSM data based on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies to rate the *amount*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Amount-Based Aggregation:

When aggregating the amount, the helper opcode assigns a reauthorization amount to the PIN_FLD_AMOUNT field in the PIN_FLD_RATING_INFO substruct. The method the opcode uses to calculate the reauthorization amount depends on the value passed in the PIN_FLD_AGGREGATE_MODE field:

- When PIN_FLD_AGGREGATE_MODE is set to **4**, the opcode uses the amount passed in the input flist.
- When PIN_FLD_AGGREGATE_MODE is set to **8**, the opcode calculates the reauthorization amount based on the PIN_FLD_RATING_MODE field:
  - When PIN_FLD_RATING_MODE is **0**, the opcode adds the amount passed in the input flist to the amount in the **/active_session** object.
  - When PIN_FLD_RATING_MODE is **1**, the opcode adds the amount passed in the input flist to the amount in the **/reservation/active** object.

For Duration-Based Requests:

When aggregating the duration, the helper opcode assigns a starting timestamp and ending timestamp to the PIN_FLD_START_T and PIN_FLD_END_T fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct.

The opcode assigns a starting timestamp based on the following:

- If PIN_FLD_START_T is passed in the input flist and the timestamp is earlier than the timestamp in the **/active_session** object, the opcode assigns the starting timestamp from the input flist.
- If PIN_FLD_START_T is passed in the input flist and the **/active_session** object does not already exist, the opcode assigns the starting timestamp from the input flist.
- If PIN_FLD_START_T is not passed in and the **/active_session** object does not already exist, the opcode assigns the starting timestamp from **pin_virtual_time**.

The opcode assigns an ending timestamp based on the following:

- If PIN_FLD_END_T is passed in the input flist, the opcode assigns the ending timestamp from the input flist.
- If PIN_FLD_QUANTITY is passed in the input flist, the opcode calculates the ending timestamp based on the values passed in the PIN_FLD_AGGREGATE_MODE field:
  - When PIN_FLD_AGGREGATE_MODE is **4**, the opcode calculates the ending timestamp by adding the starting timestamp and the quantity.
  - When PIN_FLD_AGGREGATE_MODE is **8**, the opcode populates the value based on the PIN_FLD_RATING_MODE field:
    * When PIN_FLD_RATING_MODE is **0**, the opcode calculates the ending timestamp by adding together the ending timestamp from the **/active_session** object and the quantity from the input flist.
    * When PIN_FLD_RATING_MODE is **1**, the opcode calculates the ending timestamp by adding together the starting timestamp, the reserved quantity from **/reservation/active**, and the quantity from the input flist.
- If an ending timestamp or quantity is not passed in, the opcode retrieves the default reauthorization quantity from the **/config/reserve/gprs** object. The opcode calculates the ending timestamp by adding together the default reauthorization quantity and the starting timestamp.

For Volume-Based Requests:

When aggregating the volume, the helper opcode assigns the reauthorization volume to the PIN_FLD_BYTES_UPLINK and PIN_FLD_BYTES_DOWNLINK fields in the PIN_FLD_RATING_INFO substruct. The method the opcode uses to calculate the reauthorization volume depends on the value passed in the PIN_FLD_AGGREGATE_MODE field:

- When PIN_FLD_AGGREGATE_MODE is set to **4**, the opcode assigns the volume passed in the input flist.
- When PIN_FLD_AGGREGATE_MODE is set to **8**, the opcode calculates the reauthorization volume based on the PIN_FLD_RATING_MODE field:
  - When PIN_FLD_RATING_MODE is set to **0**, the opcode adds the bytes uploaded or downloaded from the input flist to the value in the **/active_session** object.
  - When PIN_FLD_RATING_MODE is set to **1**, the opcode adds the bytes uploaded or downloaded from the input flist to the value in the **/reservation/active** object.

Preparing GSM-Specific Input Flists for Stopping Accounting Sessions

Use the PCM_OP_GSM_AAA_POL_STOP_ACCOUNTING_PREP_INPUT helper opcode to aggregate GSM data and then prepare an flist for ending a prepaid GSM session.

This helper opcode performs the following actions:

1. Compiles data from the input flist and the specified **/active_session/telco/gsm** object.
2. Determines how to aggregate the data by reading the PIN_FLD_AGGREGATE_MODE field from the input flist:
   - **1** specifies that the amount or quantity passed in the input flist is *cumulative* (that is, it represents the total amount or quantity used during the session).
   - **2** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the **/active_session** object's previous usage value.
3. Aggregates the data according to a specified value. By default, the helper opcode aggregates by duration, volume, or amount, but you can customize it to use different aggregation criteria.
4. Creates an input flist that can be passed to the appropriate Activity opcode. You can customize the information included in the input flist.

Preparing GSM-Specific Input Flists for Updating Accounting Sessions

Use the PCM_OP_GSM_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT helper opcode to aggregate GSM data and then prepare an flist for updating an existing prepaid GSM session.

This opcode aggregates GSM data by the amount passed in the input flist. If an amount is not passed in the PIN_FLD_AMOUNT field, this opcode aggregates GSM data based on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies to rate the *amount*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Amount-Based Aggregation:

When aggregating the amount, the helper opcode prepares the PIN_FLD_BALANCES array in the PIN_FLD_RATING_INFO substruct, indexed by the currency type.

- If PIN_FLD_AGGREGATE_MODE is **1**, the opcode assigns the amount from the input flist.
- If PIN_FLD_AGGREGATE_MODE is **2**, the opcode calculates the amount by adding together the amount passed in the input flist and the amount from the **/active_session** object.

For Duration-Based Requests:

When aggregating the duration, the helper opcode assigns a starting timestamp and ending timestamp to the PIN_FLD_START_T and PIN_FLD_END_T fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct.

- For the starting timestamp, the helper opcode assigns the timestamp from either the **/active_session** object or the input flist, whichever is earlier. If a starting timestamp is not present in the input flist or the **/active_session** object, the helper opcode does not modify the existing starting timestamp.
- For the ending timestamp, the helper opcode assigns the timestamp from either the **/active_session** object or the input flist, whichever is later. If an ending timestamp is not present in the input flist or the **/active_session** object, the helper opcode does not modify the existing ending timestamp.

For Volume-Based Requests:

When aggregating the volume, the helper opcode assigns the volume uploaded or downloaded by the customer to the PIN_FLD_BYTES_UPLINK or PIN_FLD_BYTES_DOWNLINK field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT.PIN_FLD_TELCO_INFO substruct. The method the helper opcode uses to calculate the volume depends on the value passed in the PIN_FLD_AGGREGATE_MODE flist field:

- When PIN_FLD_AGGREGATE_MODE is **1**, the opcode uses the bytes passed in the input flist.
- When PIN_FLD_AGGREGATE_MODE is **2**, the opcode calculates the volume by adding together the bytes passed in the input flist and the bytes in the **/active_session** object.

Building Search Templates for GSM Session Objects

Use the PCM_OP_GSM_AAA_POL_SEARCH_SESSION helper opcode to build a search template for finding **/active_session/telco/gsm** objects or **/event/session/telco/gsm** objects. By default, this helper opcode sets the search criteria to the following, but you can customize it to use other criteria:

- For **/active_session/telco/gsm** objects, the authorization ID.
- For **/event/session/telco/gsm** objects, the network session ID, MSISDN, and start time.

You specify the value for the search criteria on the input flist. For example, if the search criteria is set to authorization ID, you specify the ID to search for in the PIN_FLD_AUTHORIZATION_ID field.

Building Search Templates for GSM Active Session Objects

Use the PCM_OP_GSM_AAA_POL_ACC_ON_OFF_SEARCH helper opcode to build a search template that can be used to find an **/active_session/telco/gsm** object.

By default, this helper opcode uses the call's origination network as the search criterion, but you can customize it to use this search criteria:

- Authorization ID
- SCP name
- MSID
- Your custom criteria

You specify the value for the search criteria on the input flist. For example, if the search criteria is set to origination network, you specify which network to search for in the PIN_FLD_ORIGIN_NETWORK field.

Aggregating Return GSM Data

Use the PCM_OP_GSM_AAA_POL_POST_PROCESS helper opcode to aggregate data returned by the reauthorization process. The method used to aggregate the data depends on the value passed in the PIN_FLD_AGGREGATE_MODE flist field:

For Amount-Based Requests:

- When the field is set to **4** (aggregated mode), the helper opcode calculates the amount to return by adding the amount returned from the Activity opcode to the previously reserved amount.
- When the field is set to **8** (incremental mode), the helper opcode does not modify the return data.

For Volume-Based Requests:

- When the field is set to **4** (aggregated mode), the helper opcode does not modify the return data.
- When the field is set to **8** (incremental mode), the helper opcode calculates the quantity to return by subtracting the previously reserved quantity from the quantity returned from the Activity opcode.

For Duration-Based Requests: (when the input is in INCREMENAL MODE)

- When the field is set to **4** (aggregated mode), the helper opcode does not modify the return data.
- When the field is set to **8** (incremental mode), the helper opcode calculates the quantity to return by subtracting the previously reserved quantity from the quantity returned from the Activity opcode.

11

Installing and Configuring GSM Manager and Provisioning Data Manager

This chapter describes how to install Oracle Communications Billing and Revenue Management (BRM) GSM Manager and the Wireless Provisioning Data Manager.

Important: Before installing the GSM Manager and Wireless Provisioning Data Manager, you should be familiar with the overall integration and installation procedures. See the following documents:

- About Integrating Wireless Services
- Overview of BRM Wireless Services Installation

About the GSM Manager Components

The GSM Manager installation includes two sets of components:

- GSM server components
- Provisioning Data Manager

You can choose to install either or both of these sets of components on the same machine or on different machines.

GSM Manager Customer Center Extension adds functionality to the core Customer Center.

Note:

- To install GSM Manager Customer Center Extension, see "Installing GSM Manager Customer Center Extension on Windows" in *BRM Installation Guide*.
- The GSM Self-Care Manager pages are installed with Self-Care Manager. See "Setting Up Customer Self Care with Self-Care Manager" in *BRM Managing Customers*.

Mandatory Configuration Tasks

After you install the GSM Manager software, you must:

- Be sure that the Connection Manager (CM) and Provisioning Data Manager are configured correctly, and if not, configure them. In most cases, configuration is

done during installation, but you should verify that it was done correctly. See the following topics:

  - Configuring the Provisioning Data Manager
  - Connecting the Connection Manager to the Provisioning Data Manager

- Configure the event notification feature required for GSM provisioning. See "Configuring Event Notification for GSM Manager".
- Load the supplementary service and service-level ERA provisioning flag descriptions that Customer Center displays. See "Loading GSM Provisioning Flag Definitions".
- Load provisioning tags. See "About Provisioning Tags for Telco Services".

Hardware and Software Requirements

Before installing GSM Manager, you must install:

- Third-Party software, which includes the PERL 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 10*g* or Oracle 11*g*.

Before installing GSM Manager Customer Center Extension, you must install:

- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- Customer Center. See "Installing Customer Center on Windows" in *BRM Installation Guide*.
- (**Windows**) An application such as WinZip for extracting compressed files.

GSM Manager is supported on the HP-UX IA64, Solaris, AIX, and Linux operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

The GSM Manager Customer Center Extension is supported on the Windows platform and requires approximately 10 MB of disk space. To install GSM Manager Customer Center Extension, see "Installing GSM Manager Customer Center Extension on Windows" in *BRM Installation Guide*.

Installing GSM Manager and Provisioning Data 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 GSM 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 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.

Caution: 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 the temp_dir directory and enter this command:

7.5.0_GSM_Mgr_*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. (Optional) If you are *not* installing all GSM Manager components on this computer, use the custom install when asked to specify the setup type. Select the components you are installing by typing their respective numbers and click **Next**. The components are:
   - **GSM Manager**
   - **TCFramework**
   - **Provisioning_DM**
5. Follow the instructions displayed during installation. The default installation directory for GSM Manager is **opt/portal/7.5**.

Note: The installation program does not prompt you for the installation directory if BRM or GSM Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

6. Go to the directory where you installed the GSM Manager package and source the **source.me** file:

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

7. Go to the *BRM_Home*/**setup** directory and run the **pin_setup** script.

Your GSM Manager installation is now complete. For information on starting the Provisioning Data Manager, see "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Important: You should verify that the GSM Manager FMs were added to the Connection Manager (CM) configuration file. See "Using configuration files to connect and configure components" in *BRM System Administrator's Guide*.

Configuring and Testing GSM Manager and Provisioning Data Manager

Important: Install GSM Manager and the Provisioning Data Manager software before starting these configuration tasks.

In most cases, the server configuration tasks are handled during installation. However, you might need to manually configure some components. You should check the configuration after installation to verify that it is correct.

To configure your GSM Manager system, follow the configuration steps in these sections:

1. Applying the Correct Partitioning Layout to Event Tables
2. Configuring the Provisioning Data Manager
3. Connecting the Connection Manager to the Provisioning Data Manager
4. Creating Network Elements
5. Loading GSM Configuration Files
6. Loading the Sample GSM Price List
7. Testing GSM Provisioning

For information about configuration files, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Applying the Correct Partitioning Layout to Event Tables

Important: If your event tables are partitioned, this is a mandatory configuration task. If your event tables are not partitioned, skip this task.

When you install GSM Manager, you create additional event tables. Run the **partition_utils** utility from the *BRM_Home*/**apps/partition_utils** directory. See "Adding Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Configuring the Provisioning Data Manager

To configure the Provisioning Data Manager, you edit the Provisioning Data Manager configuration file (*BRM_Home*/**sys/dm_prov_telco/pin.conf**).

1. Open the Provisioning Data Manager configuration file (*BRM_Home*/**sys/dm_prov_telco/pin.conf**).
2. Edit the file according to the instructions in it.

   Important: You must edit the **dm_provision prov_ptr** entry to connect to the provisioning system.

   - For information about the entries specific to the Provisioning Data Manager, see "Provisioning Data Manager Configuration File Entries".
   - For information about Data Manager configuration, see "Configuring DM Front Ends and Back Ends" in *BRM System Administrator's Guide*.

   The following entries should appear in the Provisioning Data Manager **pin.conf** file and have the correct settings.

   Note:

   - These entries are more fully described in the **pin.conf** file itself.
   - If you use HP-UX or AIX, replace **solaris** with **hpux** or **aix** in these instructions.

```
- dm dm_db_no 0.0.10.2 / 0
- dm dm_logfile BRM_Home/dm_prov_telco/dm_prov_telco/dm_provision.pinlog
- dm dm_port 11990
- dm dm_sm_obj BRM_Home/sys/dm_prov_telco/dm_prov_telco.so
- dm_provision prov_ptr ip Hostname_of_the_provisioning_system20000
- dm_provision connect_retry_interval 0
- dm_provision connect_retries 0
- dm dm_trans_timeout 0
```

3. Save the file.

   Important: Before starting the Provisioning Data Manager, you must start the provisioning system.

Provisioning Data Manager Configuration File Entries

Table 11–1 lists the Provisioning Data Manager configuration file entries.

Table 11–1 Provisioning Data Manager Configuration File Entries

Entry	Description
- dm dm_db_no	Specifies the Provisioning Data Manager database number. The number is **0.0.10.2 / 0**. **Important:** The first four digits of this number (**0.0.10.2**) must match the database number in the pointer to the Provisioning Data Manager in the CM **pin.conf** file. See "Connecting the Connection Manager to the Provisioning Data Manager".
_dm dm_logfile	Specifies the path of the log file.
- dm dm_port	Specifies the port number on which the Provisioning Data Manager listens for connections. The default port number is **11990**. **Important:** This number must match the port number in the pointer to the Provisioning Data Manager in the CM **pin.conf** file. For more information, see "Connecting the Connection Manager to the Provisioning Data Manager".
- dm dm_sm_obj	Specifies a pointer to the shared library that contains the code for the Provisioning Data Manager.
- dm_provision prov_ptr ip	Specifies the host name of the provisioning system. **The default port number is 20000**.
- dm_provision connect_retry_interval	Specifies the retry interval in seconds.
- dm_provision connect_retries	Specifies the number of retries.
- dm dm_trans_timeout	Specifies the timeout interval in seconds.

Connecting the Connection Manager to the Provisioning Data Manager

The CM configuration file includes entries that point to the Provisioning Data Manager.

To check or add these entries:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) and look for the following entries:

```
- cm dm_pointer 0.0.10.2 ip Hostname_of_the_Provisioning_Data_Manager 11990
- fm_prov_telco prov_db 0.0.10.2 / 0
```

2. If the entries do not exist, add them. For more information, see "Guidelines for Database and Port-Number Entries" in *BRM System Administrator's Guide*.
3. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Creating Network Elements

Important: This is a mandatory configuration task.

By default, there are only sample network elements. Before creating telephone numbers and SIM cards, you need to add the network elements that you will use with your GSM services. Your customized network elements are displayed in SIM Administration Center and Number Administration Center.

You assign network elements to SIM cards and telephone numbers:

- When you create a block of numbers, you specify the network element for all the numbers in the block.
- When you create an order of SIM cards, you specify the network element for all the SIM cards in the order.

For more information, see the following sections:

- About Managing Telephone Numbers
- About Managing SIM Card Inventory

Note: You cannot define network elements by brand; all network elements can be used by any brand.

Loading GSM Configuration Files

This section describes how to use utilities to load configuration files required for your system.

- Loading Provisioning
- Loading Service Order States
- Loading the Telco Event Map
- Loading GSM Provisioning Flag Definitions
- Loading the Permit Mapping Files

Note: Account-level ERAs and sample provisioning tags are loaded as part of installation.

Loading Provisioning

Load provisioning by running the **load_pin_telco_provisioning** utility:

```
load_pin_telco_provisioning -dv pin_telco_provisioning
load_pin_telco_provisioning -dv pin_telco_provisioning_gsm
```

For more information, see "load_pin_telco_provisioning".

Loading Service Order States

Load telco and GSM service order states by running the **load_pin_telco_service_order_state** utility:

```
load_pin_telco_service_order_state -dv pin_telco_service_order_state
load_pin_telco_service_order_state -dv pin_telco_service_order_state_gsm
load_pin_telco_service_order_state -dv pin_telco_service_order_state_gsm_data
load_pin_telco_service_order_state -dv pin_telco_service_order_state_gsm_fax
load_pin_telco_service_order_state -dv pin_telco_service_order_state_gsm_sms
load_pin_telco_service_order_state -dv pin_telco_service_order_state_gsm_telephony
```

For more information, see "load_pin_telco_service_order_state".

Loading the Telco Event Map

> **Note:** If you use custom events you must manually edit the *BRM_Home*/**sys/data/config/pin_event_map_telco_gsm** file.

Load the telco event mapping file by running the **load_event_map** utility, as in this example:

```
load_event_map -dv pin_event_map_telco_gsm
```

For more information, see **load_event_map** in *BRM Setting Up Pricing and Rating*.

Loading GSM Provisioning Flag Definitions

> **Important:** This is a mandatory configuration task.

Customer Center displays the provisioning flags of GSM supplementary services and service-level ERAs. You can customize and localize the descriptions for these flags. For information about these flags, see "About Provisioning GSM Services".

To add or change provisioning flags in the BRM database, you edit a copy of the **telco_features_and_profiles_states.en_US** sample file in the *BRM_Home*/**sys/msgs/telcofeaturesandprofilesstates** directory. 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 telco_features_and_profiles_states.locale
```

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

> **Note:** 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*.

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

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

Loading the Permit Mapping Files

Load the GSM number and SIM permit mapping files by running the **load_pin_device_permit_map** utility:

```
load_pin_device_permit_map pin_device_permit_map_num_telco_gsm
load_pin_device_permit_map pin_device_permit_map_sim_telco_gsm
```

For more information on this utility, see "Defining Device-to-Service Associations" in *BRM Developer's Guide*.

Configuring Event Notification for GSM Manager

Important: This is a mandatory configuration task.

To enable provisioning, GSM delayed activation, and communication with Pipeline Manager, BRM uses event notification.

Before you can use GSM Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them. See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Ensure that the merged file includes the entire event notification list in the *BRM_Home*/**sys/data/config/pin_notify_telco** file.
3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list. See "Editing the Event Notification List" in *BRM Developer's Guide*.
4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger. See "Triggering Custom Operations" in *BRM Developer's Guide*.
5. Load your final event notification list into the BRM database. See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Loading the Sample GSM Price List

Caution: Loading the sample price list overwrites existing plan lists. To keep existing plan lists, you need to open the sample GSM price list and add the GSM plans to your existing plan lists.

When you install GSM Manager, a sample GSM price list is copied to your system. You can open it directly in Pricing Center, but you must load it into the BRM database to create GSM accounts with it.

To load the sample GSM price list:

1. Start Pricing Center.
2. Choose **File** - **Import - Real-time Data**.
3. Import the **TelcoGSMSamplePricePlan.xm**l file on your computer.

   The default location on Windows is one of the following:

   - If you installed Pricing Center with Java Web Start, the default location is your Windows **My Document** folder.
   - If you installed Pricing Center as a standalone application, the default location is **C:\Program Files\Portal Software\PricingCenter\Sample_Price_Plans\Optional_Manager_Plans**.

Note: As part of importing the XML file, you either confirm your current database connection or specify a database.

The objects in the price list appear in the Snapshot pane on the left side of the Pricing Center window.

4. Drag the pricing objects from the Snapshot pane to the price list work area.
5. Choose **File** - **Commit to Portal Database** and click **OK** in the confirmation box.
6. Enter your BRM database login information and click **Connect.**

   The pricing objects in the sample price list are added to your BRM database.
7. After the sample plan list is committed to the database, click **OK** in the confirmation box.

Loading the Sample Price List XML File

Instead of using Pricing Center, you can use the **loadpricelist** utility to load the **TelcoGSMSamplePricePlan.xml** file. This file includes the same price list as the IPL file. By default, the file is installed in *BRM_Home***/setup/scripts**.

For information, see the following sections in *BRM Setting Up Pricing and Rating*:

- **loadpricelist**
- Using the XML Pricing Interface to Create a Price List

Testing GSM Provisioning

To test GSM provisioning, you use the BRM Network Simulator utility. See "Testing Provisioning Using BRM Network Simulator".

Uninstalling GSM Manager and Provisioning Data Manager

To uninstall GSM Manager and Provisioning Data Manager, run the *BRM_Home***/uninstaller/GSM_Mgr/uninstaller.bin**.

12

Installing GSM AAA Manager

This chapter describes how to install Oracle Communications Billing and Revenue Management (BRM) GSM AAA Manager.

Before installing GSM AAA Manager, you should be familiar with the overall integration and installation procedures related to wireless services. See the following documents:

- About Integrating Wireless Services
- Overview of BRM Wireless Services Installation

Hardware and Software Requirements

You can install GSM AAA Manager on HP-UX IA64, Solaris, AIX, and Linux operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Before installing GSM AAA Manager, you must install and configure:

- Third-Party software, which includes the PERL 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 10*g* or Oracle 11*g*.
- GSM Manager 2.0. See "Installing and Configuring GSM Manager and Provisioning Data Manager".
- Resource Reservation Manager. See "Installing Resource Reservation Manager" in *BRM Configuring and Collecting Payments*.

Installing GSM AAA 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 GSM AAA Manager, perform these steps:

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 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.

   Caution: 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 the temp_dir directory and enter this command:

   ```
   7.5.0_GSM_AAA_Mgr_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 GSM AAA Manager is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or GSM AAA Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the GSM AAA 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.

7. If your event tables are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home*/**apps/partition_utils** directory:

   ```
   perl partition_utils.pl -o update
   ```

   See "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Your GSM AAA Manager installation is now complete.

Uninstalling GSM AAA Manager

To uninstall GSM AAA Manager, run the *BRM_Home*/**uninstaller/GSM_AAA_Mgr/uninstaller.bin**.

13

Setting Up GSM Wireless Pricing

This chapter describes how to create a GSM price list. It describes how Pipeline Manager and Oracle Communications Billing and Revenue Management (BRM) pricing components work together, and describes the GSM Manager sample price list.

- For information about creating a price list, see "About Creating a Price List" in *BRM Setting Up Pricing and Rating*.
- For information about setting up rate plans for pipeline rating, see "About Pipeline Rating" in *BRM Configuring Pipeline Rating and Discounting*.

Important: Before using the sample GSM price list, you need to load it into the BRM database.

About Creating GSM Products

How you define products depends mostly on how you plan to offer GSM services, supplementary services, and ERAs, all of which are defined in products by using provisioning tags. For more information, see "About Provisioning Tags for Telco Services".

In addition, you should consider that you manage supplementary services by changing product status. For example, to inactivate call forwarding you inactivate the product it was purchased with. However, if other supplementary services or service-level ERAs are bundled in the same product, all services and ERAs purchased with that product are also inactivated.

Specifying Usage Rate Plan Names

When selecting a BRM rate plan, the pipeline rating module chooses a rate plan based on the account's product, service, and the type of event. If there is more than one rate plan available, the rating module chooses the first one it finds that matches the criteria. Therefore, you should do one of the following:

- Define only one usage product per service.
- Use the same rate plan name in all usage products for the same service.
- Create rate plans for multiple products for the same types of customers. For example, the sample price list includes only four rate plans, one for each type of plan.

For more information, see "About Pipeline Rating" in *BRM Configuring Pipeline Rating and Discounting*.

About Resources

In BRM, use resources as you normally do for creating rate plans. See "About Resources" in *BRM Setting Up Pricing and Rating*.

For pipeline usage rate plans, the resource is not used by real-time rating. However, you need to specify the correct resource to allow BRM to update the account balance.

In Pipeline Manager, you need to configure the same resource by configuring the FCT_BillingRecord module in the registry.

About RUMs

You define RUMs for usage rates in the pipeline pricing model.

By default, the GSM usage RUMs are Message, Size, Uplink, and Downlink. These RUMs are used for real-time rating. For batch rating using Rated Event Loader and delayed events, the only RUM used is Occurrence.

For cycle, purchase, and cancel rates, use the Occurrence RUM, as you normally would.

About Impact Categories

You do not need to synchronize impact categories between pipeline rating and real-time rating. You define impact categories in the Pipeline Manager database; real-time rate plans do not need them when you use pipeline rating.

About the Event Map

To rate usage events, you can specify which events are rated for each service. See "About Specifying the Events to Rate in a Product" in *BRM Setting Up Pricing and Rating*. The event used for rating GSM services is **/event/delayed/session/gsm**. This is the event that is loaded into the BRM database by RE Loader.

About the Sample GSM Pricing Configuration

You can use the sample GSM pricing for two purposes:

- To learn about how GSM rating and pricing works.
- As a starting place for you pricing.

The sample GSM pricing is defined in two components:

- The BRM price list includes the plans that you use for creating accounts, and the deals and products that customers purchase and own. You open the BRM sample GSM price list in Pricing Center. There are two versions of the sample price list:
  - **GSMSamplePricePlan.xml**
  - **GSMSamplePricePlan.ipl**
- The Pipeline Manager pricing model defines the rate plans, zone maps, and impact categories that are used for rating usage events.

About the Sample GSM G/L IDs

The default **pin_glid** file includes the following sample general ledger IDs in Table 13–1.

Table 13–1 Sample File G/L IDs

G/L ID	Description
1420	National airtime telephony
1421	International airtime telephony
1430	National airtime SMS
1431	International airtime SMS
1441	International airtime other GSM
1450	National usage for GPRS
1451	International usage for GPRS
1460	National usage for other services
1461	International usage for other services
1500	Euro incoming roaming GSM
1501	Euro outgoing roaming GSM
1503	International incoming roaming GSM
1504	International outgoing roaming GSM
1510	National value added services
1511	International value added services

For more information, see "About Collecting General Ledger Data" in *BRM Collecting General Ledger Data*.

About the Sample GSM Price List

The sample GSM price list includes the following plans and deals:

- Corporate Plus Data Add-On Plan
- Corporate Plus Fax Add-On Plan
- Corporate Plus GSM Plan
- Standard Data Add-On Plan
- Standard Fax Add-On Plan
- Standard GSM Plan
- Teen Data Add-On Plan
- Teen Fax Add-On Plan
- Teen GSM Plan
- Corporate Plus Telephony Add-On Deal
- Standard Telephony Add-On Deal
- Teen SMS Add-On Deal
- Teen Telephony Add-On Deal
- Settlement Plan

All of these plans and deals (except for the Settlement plan) share the following attributes:

Note: See the Settlement plan for settlement-specific details.

- The currency resource is the euro (currency code 978).
- The event that is rated by usage events is **/event/delayed/session/telco/gsm**.
- Usage rate plans are defined, including rate plans. These usage rate plans are not used for rating. The rate plans that are used for rating are defined in the pipeline rate plans.

To make the price list easier to understand, there are no discounts, there are no tax codes, and all products are always valid. In addition, all deals include only one product.

Corporate Plus Data Add-On Plan

Use this plan to add the GSM data service (**/service/telco/gsm/data**) to an existing account. The fees defined by this plan are:

- 20 euro per month subscription fee.
- A usage rate plan for the GSM data service. The amount charged is defined by the pipeline rate plan.

This plan includes one deal, which includes one product.

- Corporate Plus Data Add-on deal
- Corporate Plus Data Add-on product

The service for the Corporate Plus Data Add-on product is **/service/telco/gsm/data**.

This product includes the following:

- A monthly cycle forward fee of 20 euro.
- The CorpPlus usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Data Corporate Plus provisioning tag. This provisioning tag includes only the GSM data bearer service, specifically, the duplex asynchronous 9600bps PAD access.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE B46

Corporate Plus Fax Add-On Plan

Use this plan to add the GSM fax service (**/service/telco/gsm/fax**) to an existing account. The fees defined by this plan are:

- 10 euro per month subscription fee.
- A usage rate plan for the GSM fax service. The amount charged is defined by the pipeline rate plan.

This plan includes one deal, which includes one product.

- Corporate Plus Fax Add-on deal
- Corporate Plus Fax Add-on product

Corporate Plus Fax Add-On Product

The service for the Corporate Plus Fax Add-on product is **/service/telco/gsm/fax**.

This product includes the following:

- A monthly cycle forward fee of 10 euro.
- The CorpPlus usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Fax Corporate Plus provisioning tag. This provisioning tag supports only the GSM fax bearer service, specifically, Automatic Facsimile Group 3.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE T62

Corporate Plus GSM Plan

This plan includes the following charges:

- 200 euro per month for SMS and telephony services.
- Usage rate plans for SMS and telephony. The amount charged is defined by the pipeline rate plan.

The services included are:

- **/service/telco/gsm/sms**
- **/service/telco/gsm/telephony**

The supplementary services are:

- Caller ID
- Call waiting
- Call blocking (BAICR and BIOC)
- Unconditional call forwarding
- Voice mail
- Auto roam

This plan includes two deals, one for the telephony service and one for the SMS service. Each deal includes only one product.

- The Corporate Plus SMS deal includes the Corporate Plus SMS product.
- Corporate Plus Telephony deal includes the Corporate Plus Telephony product.

Corporate Plus SMS Product

The service for the Corporate Plus SMS product is **/service/telco/gsm/sms**.

This product includes the following:

- The CorpPlus usage rate plan. The amount charged is defined by the pipeline rate plan.
- The SMS Corporate Plus provisioning tag. The provisioning tag supports:
  - The SMS bearer service.
  - Supplementary services to allow for inter-carrier access.

  The provisioning tag contains these entries:

- Service extension: PIN_FLD_BEARER_SERVICE T20
- Features: BAOC, BICRO

Corporate Plus Telephony Package Product

The service for the Corporate Plus Telephony Package is **/service/telco/gsm/telephony**.

This product includes the following:

- A cycle forward rate plan that charges 200 euro per month.
- The CorpPlus usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Voice Corporate Plus provisioning tag. The provisioning tag supports:
  - The T11 telephony service.
  - Supplementary services for caller ID, call waiting, call blocking (BAICR and BIOC), unconditional call forwarding, voice mail, and auto roam.

  The provisioning tag contains these entries:
  - Service extension: PIN_FLD_BEARER_SERVICE T11
  - Features: CLIP, CW, BAICR, BOIC, CFU, VMBOX, ROAM

Standard Data Add-On Plan

Use this plan to add the GSM data service (**/service/telco/gsm/data**) to an existing account. The fees defined by this plan are:

- 100,000 free bytes (100 free Kilobytes) per month.

  Important: To match the resources defined in Pipeline Manager, this resource must be configured as bytes, not Kilobytes.

- 20 euro per month subscription fee.
- A usage rate plan for the GSM data service. The amount charged is defined by the pipeline rate plan.

This plan includes one deal, which includes one product.

- Standard Data Add-on deal
- Standard Data Add-on product

The service for the Standard Data Add-on product is **/service/telco/gsm/data**.

This product includes the following:

- A monthly cycle forward fee of 20 euro
- The Standard usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Data Standard Add-on provisioning tag. This provisioning tag includes only the GSM data bearer service, specifically, the duplex asynchronous 9600bps PAD access.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE B46

Standard Fax Add-On Plan

Use this plan to add the GSM fax service (**/service/telco/gsm/fax**) to an existing account. The fees defined by this plan are:

- 10 euro per month subscription fee.
- A usage rate plan for the GSM fax service. The amount charged is defined by the pipeline rate plan.

This plan includes one deal, which includes one product.

- Standard Fax Add-on deal
- Standard Fax Add-on product

Standard Fax Add-On Product

The service for the Corporate Plus Fax Add-on product is **/service/telco/gsm/fax**.

This product includes the following:

- A monthly cycle forward fee of 10 euro.
- The Standard usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Fax Standard Add-on provisioning tag. This provisioning tag supports only the GSM fax bearer service, specifically, Automatic Facsimile Group 3.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE T62

Standard GSM Plan

This plan includes the following charges and credits:

- 50 euro per month for telephony and SMS access.
- 3600 free seconds (60 free minutes) per month.

  Important: To match the resources defined in Pipeline Manager, this resource must be configured as seconds, not minutes.

- Usage rate plans for GSM telephony and SMS. The amount charged is defined by the pipeline rate plan.

There are no folds to handle unused seconds or bytes because Pipeline Manager calculates the rollover amounts.

The services are:

- **/service/telco/gsm/sms**
- **/service/telco/gsm/telephony**

The supplementary services are:

- Caller ID
- Call waiting
- Voice mail

This plan also includes the friends and family ERA.

This plan includes two deals, each of which includes one product:

- The Standard SMS deal includes the Standard SMS product.
- The Standard GSM Telephony deal includes the Standard GSM Telephony product.

Standard SMS Product

The service for the Standard SMS product is **/service/telco/gsm/sms**.

This product includes the following:

- The Standard rate plan. The amount charged is defined by the pipeline rate plan.
- The SMS Standard provisioning tag. The provisioning tag supports:
  - The T20 SMS bearer service.
  - Supplementary services that support inter-carrier access.

  The provisioning tag contains these entries:
  - Service extension: PIN_FLD_BEARER_SERVICE T20
  - Features: BAOC, BICRO

Standard GSM Telephony Product

The service for the Standard GSM Telephony product is **/service/telco/gsm/telephony**.

This product includes the following:

- A monthly cycle forward rate plan that supports a 50 euro per month subscription charge.
- The Standard usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Voice Standard provisioning tag. This provisioning tag supports the following:
  - The T11 telephony bearer service.
  - Supplementary services: caller ID, call waiting, voice mail.
  - The friends and family service-level ERA.

  The provisioning tag contains these entries:
  - Service extension: PIN_FLD_BEARER_SERVICE T11
  - Features: CLIP, CW, VMBOX
  - Extended rating attribute: FRIENDS_FAMILY, provisioning not required

Teen Data Add-On Plan

Use this plan to add the GSM data service (**/service/telco/gsm/data**) to an existing account. The fees defined by this plan are:

- 20 euro per month subscription fee.
- A usage rate plan for the GSM data service. The amount charged is defined by the pipeline rate plan.

This plan includes one deal which includes one product.

- Teen Data Add-on deal

- Teen Data Add-on product

The service for the Teen Data Add-on product is **/service/telco/gsm/data**.

This product includes the following:

- A monthly cycle forward fee of 20 euro.
- The Teen usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Data Teen Add-on provisioning tag. This provisioning tag includes only the GSM data bearer service, specifically, the duplex asynchronous 9600bps PAD access.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE B46

Teen Fax Add-On Plan

Use this plan to add the GSM fax service (**/service/telco/gsm/fax**) to an existing account. The fees defined by this plan are:

- 10 euro per month subscription fee.
- A usage rate plan for the GSM fax service. The amount charged is defined by the pipeline rate plan.

This plan includes one deal, which includes one product.

- Teen Fax Add-on deal
- Teen Fax Add-on product

Standard Fax Add-On Product

The service for the Corporate Plus Fax Add-on product is **/service/telco/gsm/fax**.

This product includes the following:

- A monthly cycle forward fee of 10 euro.
- The Teen usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Fax Teen Add-on provisioning tag. This provisioning tag supports only the GSM fax bearer service, specifically, Automatic Facsimile Group 3.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE T62

Teen GSM Plan

This plan includes the following charges:

- 50 euro per month for telephony and SMS access.
- Usage rate plans for GSM telephony and SMS. The amount charged is defined by the pipeline rate plan.

The services are:

- **/service/telco/gsm/sms**
- **/service/telco/gsm/telephony**

The supplementary services are:

- Caller ID
- Call waiting
- Voice mail

This plan includes two deals, each of which includes one product:

- The SMS Teen deal includes the Teen SMS product.
- The Teen Telephony deal includes the Teen Telephony product.

Teen SMS Product

The service for the Teen SMS product is **/service/telco/gsm/sms**.

This product includes the following:

- The Teen rate plan. The amount charged is defined by the pipeline rate plan.
- The SMS Teen provisioning tag. The provisioning tag supports the T20 SMS bearer service.

  The provisioning tag contains this entry:

  Service extension: PIN_FLD_BEARER_SERVICE T20

Teen Telephony Product

The service for the Teen Telephony product is **/service/telco/gsm/telephony**.

This product includes the following:

- A monthly cycle forward rate plan that supports a 2 euro per month subscription charge.
- The Teen usage rate plan. The amount charged is defined by the pipeline rate plan.
- The Voice Teen provisioning tag. This provisioning tag supports the following:
  - The T11 telephony bearer service.
  - Supplementary services: caller ID, call waiting, voice mail.

  The provisioning tag contains these entries:
  - Service extension: PIN_FLD_BEARER_SERVICE T11
  - Features: CLIP, CW, VMBOX

Corporate Plus Telephony Add-On Deal

Use this deal to add conference calling and the friends and family promotion to an account that already owns **/service/telco/gsm/telephony**.

This deal includes the Corporate Plus Telephony Add-on product. This product includes:

- A 2 euro per month subscription fee.
- The Voice Add On Teen: Promotions and SS provisioning tag. This provisioning tag supports the following:
  - Call Barring (BAICR and BIOC).
  - Call Forwarding Unconditional.
  - Auto Roam.

  - The Home Cell service ERA for the telephony service.

  The provisioning tag contains these entries:

  - Features: BAICR, BOIC, CFU, ROAM
  - Extended rating attribute: HOME_CELL, provisioning required

Standard Telephony Add-On Deal

Use this deal to add the following to an account that already owns **/service/telco/gsm/telephony**:

- Call blocking
- Unconditional call forwarding
- Roaming
- Home Cell promotion

This deal includes the Corporate Plus Telephony Add-on product. This product includes:

- A 5 euro per month subscription fee.
- The Voice Add On Standard: Promotions and SS provisioning tag. This provisioning tag supports the following:
  - Call blocking (BAICR and BIOC).
  - Call forwarding unconditional.
  - Auto roam.
  - The Home Cell service ERA for the telephony service.

  The provisioning tag contains these entries:

  - Features: BAICR, BOIC, CFU, ROAM
  - Extended rating attribute: HOME_CELL, provisioning required

Teen SMS Add-On Deal

Use this deal to add call blocking to an account that already owns **/service/telco/gsm/sms**:

This deal includes the Teen SMS Add-on product. This product includes:

- A 5 euro per month subscription fee.
- The SMS Teen Add On: Promotions and SS provisioning tag. This provisioning tag supports call blocking (BAICR and BIOC).

  The provisioning tag contains this entry:

  Features: BAOC, BICRO

Teen Telephony Add-On Deal

Use this deal to add the following to an account that already owns **/service/telco/gsm/telephony**:

- Call blocking (BAICR and BIOC).
- Call forwarding unconditional.

- Auto roam.
- The Home Cell service ERA for the telephony service.

This deal includes the Teen Telephony Add-on product. This product includes:

- 5 euro per month subscription fee.
- The Voice Add On Teen: Promotions and SS provisioning tag. This provisioning tag supports the following:
  - Call blocking (BAICR and BIOC).
  - Call forwarding unconditional.
  - Auto roam.
  - The Home Cell service ERA for the telephony service.

  The provisioning tag contains these entries:
  - Features: BAICR, BOIC, CFU, ROAM
  - Extended rating attribute: HOME_CELL, provisioning required

Settlement Plan

Use this plan to add the settlement service (**/service/settlement**) to an account.

The settlement service is required when you create accounts for network operators to collect settlements for roaming fees. The network operators are service providers with which you make roaming agreements.

This plan has the following attributes:

- The currency resource is US Dollars (currency code 840).
- There are no ratable usage events for this plan. Use the settlement plan for collecting wireless roaming settlements from roaming partners, not for rating wireless usage.
- There are no fees defined by this plan.
- The defined rate plan is not used for rating. You define a pipeline rate plan for each roaming partner (network operator) when you set up roaming in Pipeline Manager. For more information, see "About Rating Roaming Events" in *BRM Configuring Roaming in Pipeline Manager*.

This plan includes one deal, which includes one product.

- Settlement deal
- Settlement product

Settlement Deal

Use this deal to add settlement support to an account that owns the settlement service (**/service/settlement**).

Settlement Product

The service for the Settlement product is **/service/settlement**.

This product allows settlement amounts to be applied to roaming partner accounts.

Part IV

Services Framework Overview

Part IV provides an overview of the Oracle Communications Billing and Revenue Management (BRM) Services Framework. It contains the following chapters:

14

Understanding the Services Framework

This chapter provides a conceptual overview of Oracle Communications Billing and Revenue Management (BRM) Services Framework and explains how to use it to implement prepaid services.

Before reading this chapter, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

About the Services Framework

The Services Framework is a collection of opcodes, storable classes, and utilities that allows you to quickly develop BRM support for any wireless or wireline service, such as GSM and CDMA.

You can use The Services Framework to implement the following functionality:

- Collect information about prepaid customers. See "About Collecting Information for Prepaid Customers".
- Manage prepaid services. See "About Service Management".
- Provision prepaid services. See "About Provisioning Services".
- Process authentication, authorization, and accounting (AAA) requests for prepaid services. See "About Processing AAA Requests for Prepaid Services".

About Collecting Information for Prepaid Customers

The *Services Framework Manager client* includes panels in Customer Center dedicated to configuring services for customer accounts.

When a customer support representative (CSR) selects a service for a customer account, the service panel displays deal and login information, plus customized subpanels for supplementary services, devices, and extended rating attributes (ERAs).

You customize the Services Framework Manager client by using tools in the Customer Center SDK, including BRM Configurator and JBuilder.

For information about using and customizing the Services Framework Manager client, see "About Customizing the Services Framework Manager Client".

About Service Management

Service management supports life-cycle management of prepaid services. For example, it activates and deactivates services when products are purchased or canceled.

To support life-cycle management of prepaid services, prepaid service management performs the following:

- Manages prepaid service real-time and scheduled (deferred) actions for features and promotions.
  - If the service activation or deactivation date is set in the future for the product or deal that a subscriber is purchasing, a schedule object is stored in the BRM database.
  - If the date is set to the current time, service provisioning is triggered immediately.
- Manages prepaid service status based on account management actions, including:
  - Updating service and ERA objects that are impacted when the product provisioning status changes.
  - Propagating service object Close and Suspend status changes when they have an unprovisioning impact on the supplementary services and ERAs associated with a service.

For more information about managing prepaid services, see "About Managing Prepaid Services and Extended Rating Attributes".

For information about adding new prepaid services, see "Adding New Prepaid Services".

About Provisioning Services

Service provisioning allows you to notify external networks when the status of a prepaid service or device changes; for example, when a service is activated or deactivated.

When the status changes, service provisioning does the following:

1. Creates a service order, which provides information about the service and the provisioning action required.
2. Sends the service order to the external network through the Services Provisioning Data Manager (DM), **dm_prov_telco**.
3. Updates the service order's status.
4. Updates the status of the service and device associated with the service order.

You set up your system to provision custom service types by configuring service provisioning. For more information, see "About Provisioning Telco and Non-Telco Services".

About Processing AAA Requests for Prepaid Services

Services Framework AAA Manager allows you to process AAA requests for prepaid services.

When a prepaid customer attempts to use a service, the external network uses Services Framework AAA Manager to:

- Verify the customer's identity.
- Determine whether the customer is allowed to use the service by verifying that the customer owns the service and has enough resources in the account balance.
- Record information about the prepaid usage.

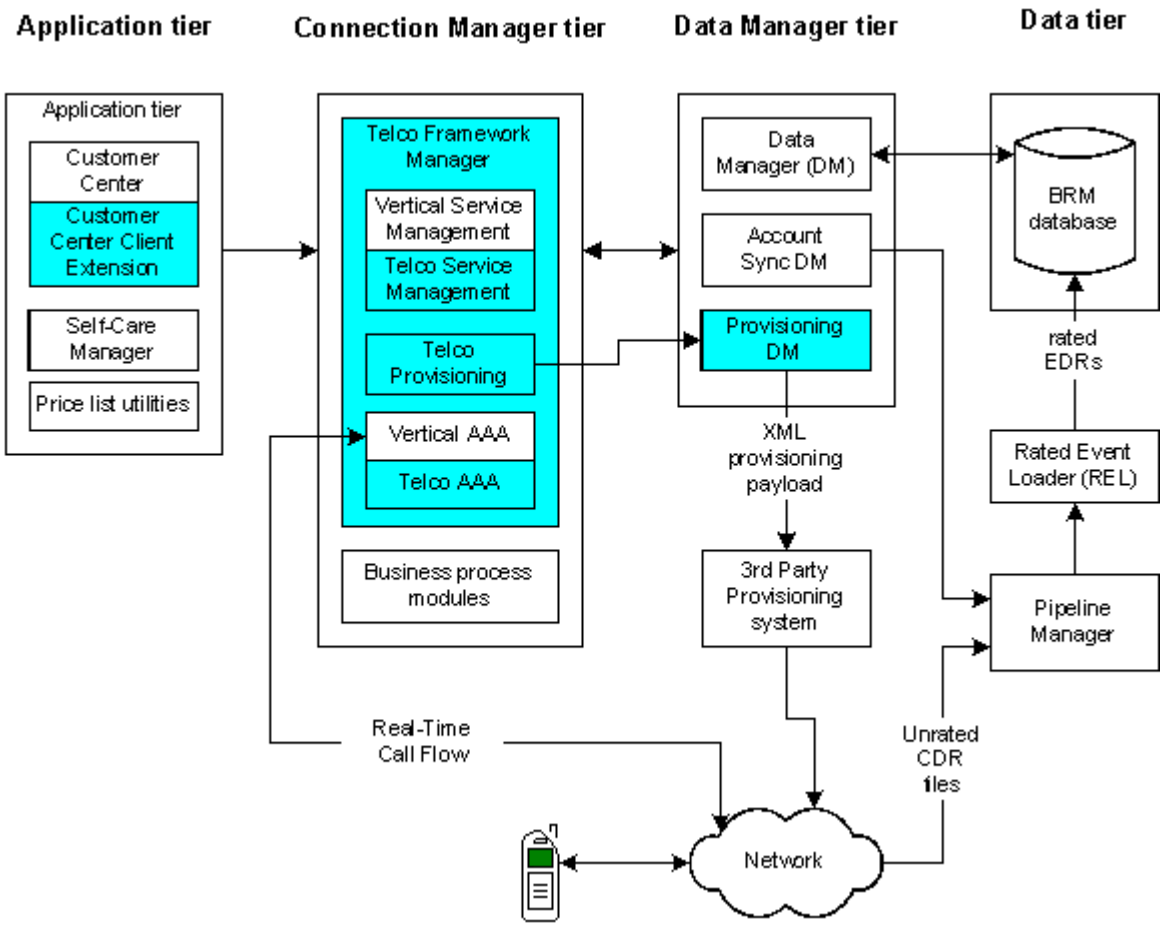
- Set up and update the provisioning policy for the session based on the subscriber's total usage of the service in association with the offer profile for the purchased plan.

You set up your system to perform AAA for custom service types by using the Services Framework AAA API. For more information, see "About Performing AAA for Prepaid Services".

Services Framework Architecture

Figure 14–1 shows an overview of the Services Framework and other BRM components in a system that supports prepaid services. Components in blue are part of the Services Framework:

Figure 14–1 Services Framework Architecture



15

Installing Services Framework Manager

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) Services Framework Manager software.

Before you read this chapter, you should be familiar with BRM concepts and architecture. See *BRM Concepts* and "Understanding the Services Framework" for more information.

Important: Services Framework Manager is an optional feature that requires a separate license.

System Requirements

Services Framework Manager is available for the HP-UX IA64, Linux, Solaris, and AIX operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Software Requirements

Before installing Services Framework Manager, you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM software. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- Oracle database software.

Installing Services Framework Manager

To install Services Framework 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.

1. Download the software to a temporary directory *(temp_dir)*. For more information on downloading the software, see "Downloading the BRM Applications Software" in *BRM Installation Guide*.

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 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.

Caution: 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 the temp_dir directory and enter this command:

```
7.5.0_BRM_Services_Framework_Mgr_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 Services Framework Manager is **/opt/portal/7.5**.

Note: The installation program does not prompt you for the installation directory if BRM or Services Framework Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the *BRM_Home***/setup** directory and run the **pin_setup** script.

Note: The **pin_setup** script starts all required BRM processes.

6. If your event tables are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home*/**apps/partition_utils** directory:

```
perl partition_utils.pl -o update
```

For more information, see "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Your Services Framework Manager installation is now complete.

Uninstalling Services Framework Manager

To uninstall Services Framework Manager, run the *BRM_Home*/**uninstaller/TelcoFrameworkMgr/uninstaller.bin**.

16

Installing Services Framework AAA Manager

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) Services Framework AAA Manager software.

Before you read this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts* and "Understanding the Services Framework" for more information.

Important: Services Framework AAA Manager is an optional feature that requires a separate license.

System Requirements

Services Framework AAA Manager is available for the HP-UX IA64, Linux, and Solaris operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Software Requirements

Before installing Services Framework AAA Manager, you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM software. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- Oracle database software.

Installing Services Framework AAA Manager

To install Services Framework AAA 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.

1. Download the software to a temporary directory (*temp_dir*). For information on downloading the software, see "Downloading the BRM Applications Software" in *BRM Installation Guide*.

Important: 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 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.

   Caution: 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 the temp_dir directory and enter this command:

   ```
   7.5.0_BRM_Services_Framework_Mgr_AAA_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 Services Framework AAA Manager is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or Services Framework Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the Services Framework AAA 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.

7. If your event tables are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home***/apps/partition_utils** directory:

   ```
   perl partition_utils.pl -o update
   ```

   For more information, see "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Your Services Framework AAA Manager installation is now complete.

Uninstalling Services Framework AAA Manager

To uninstall Services Framework AAA Manager, run the *BRM_Home***/uninstaller/TelcoFrameworkMgr_AAA/uninstaller.bin.**

17

About Customizing the Services Framework Manager Client

This chapter describes how to customize the Oracle Communications Billing and Revenue Management (BRM) Services Framework Manager client.

For information about configuring other components of Customer Center and for building and deploying Customer Center customizations, see "Customizing the Customer Center Interface" in *BRM Developer's Guide*.

About Customizing the Services Framework Manager Client

You can customize Customer Center prepaid components according to your business requirements.

To customize prepaid components of Customer Center:

- Use JBuilder to extend or create:
  - A service panel for a prepaid service.
  - One or more device panels for each service.
  - Search entry and search results panels for each device.
- For each prepaid service, you use BRM Configurator to specify:
  - The text to be displayed in Customer Center for the service panel.
  - If the default or a custom service panel should be used.
  - (**For the default service panel**) if the default extended rating attribute (ERA) panel should be displayed, and if so, specify its name.
  - (**For the default service panel**) if the supplementary services panel should be displayed, and if so, if it should be displayed in expanded mode.
  - (**For the default service panel**) the order to display devices configured for the service.
  - (**For a custom service panel**) the custom panel class name.
- For each device supported by a prepaid service, you use Configurator to specify:
  - The displayed name of the device.
  - If Customer Center should add the device to the account search.
  - If Customer Center should use system inventory retrieval to prepopulate device fields when a device is selected.

- The class name for the device panel.
- The class name for the device search panel.
- The class name for the device search results panel.

Overview of Customizing the Services Framework Manager Client

Follow these steps to configure Customer Center prepaid components:

1. Extend or create service, device, device search entry, and device search results panels by using JBuilder. See "Creating Custom Service and Device Panels".
2. Specify layout attributes and the any extended panels by using Customer Center SDK Configurator. See "Configuring Service and Device Panel Layouts by Using Configurator"

Creating Custom Service and Device Panels

You use JBuilder for these tasks:

- Creating Custom Service Panels
- Creating Custom Device Panels
- Creating Custom Device Search Panels
- Creating Custom Device Search Entry Panels
- Creating Custom Device Search Results Panels

Coding Your Customizations

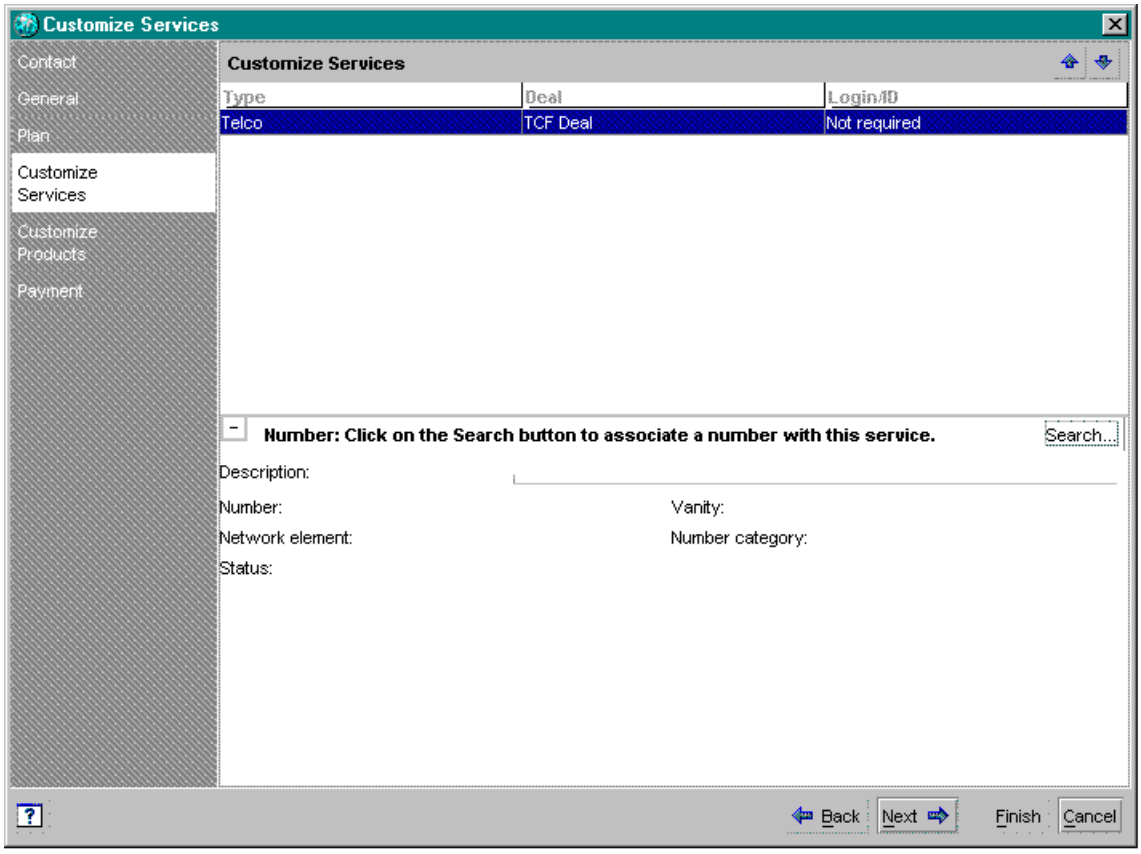
You create or extend service or device panels by using BRM Developer Center and JBuilder. See "Setting Up JBuilder to Customize the Customer Center Interface" and "Adding Custom Fields to Customer Center" in *BRM Developer's Guide*.

For panel extension code samples, see the **CustomerCareSDK\CCExamples\TelcoFramework** directory.

Creating Custom Service Panels

You can customize the default service panel by extending the base class (**PTelcoServicePanel**). By default, this base service panel is displayed when a service is selected in Customer Center by a customer support representative (CSR) as shown in Figure 17–1:

Figure 17–1 Base Service Panel



You can create a more specific panel for **/service/telco/***service_name* by extending the default panel (**PTelcoServicePanel**) and renaming it **P***service_name***Panel**.

By default, the following methods return the values specified in properties entries in the *CCSDK_home*/**CustomerCareSDK/CustCntr/custom/bin/WizardCustomizations.properties** file. This properties file is created by the Customer Center SDK Configurator when you save your changes. Table 17–1 lists the Wizard Customizations properties in the file.

Note:

- For more information on Configurator, see "Configuring Service and Device Panel Layouts by Using Configurator" and "Customizing the Customer Center Interface" in *BRM Developer's Guide*.
- For more information on these methods, see the JavaDocs in *CC SDK_home*/**CustomerCareSDK/docs** directory.

Table 17–1 Wizard Customization Properties

Method	Properties File Entry (WizardCustomizations.properties)
protected boolean isSupplementaryVisible ()	**Telco.service.***service_name***.supplementary.display = true**

Table 17–1 (Cont.) Wizard Customization Properties

Method	Properties File Entry (WizardCustomizations.properties)
protected Boolean isSupplementaryExpanded ()	**telco.service.***service_name***.supplementary.expanded = true**
protected String getExtendedAttrClassName ()	**telco.service.***service_name***.extended**
Protected List getDeviceSequence ()	**telco.service.***service_name***.devices = num bar**
protected Map getDevicePanels ()	**extended.device.bar = PBarPanel**
protected String getServiceName ()	**extended.service.***service_name* **= P***service_name***Panel**

Creating Custom Device Panels

You extend the **PTelcoDevicesBase** class to customize the layout of the device panel for a specific device. For information about the public methods in this class, see **PTelcoDevicesBase** in the Customer Center *JavaDocs*.

The default base device panel (**PDefaultDevicePanel**) appears as shown in Figure 17–2:

Figure 17–2 Default Base Device Panel

Important:]You *must* extend **TTelcoDevicesBase** because it does not contain a default user interface (UI) implementation.

Sample Device Panel Subclass Template

Customer Center SDK includes the subclass template *ccsdk_home*/**CustomerCareSDK/CCExamples/TelcoFramework/TelcoDeviceTemplate.txt**. Use this as a starting point to create a device panel subclass from **PTelcoDevicesBase**.

This template includes:

- A header.
- An **Update** button, which connects to an action listener that invokes the method **prepopulate ()**.

  Note: The **Update** button is displayed only when the prepopulate option is enabled. See "About Device Prepopulation".

- A **Search** button, which connects to an action listener that invokes the method **searchForDevice ()**, which displays the device search dialog.

- A **History** button, which connects to an action listener that invokes the method **showDeviceHistory ()**.

> **Note:** The **History** button is displayed only during account maintenance actions.

- One field each for Description, Device ID, and Status.

You can keep or remove the header and buttons, and you can append components to the panel by using JBuilder.

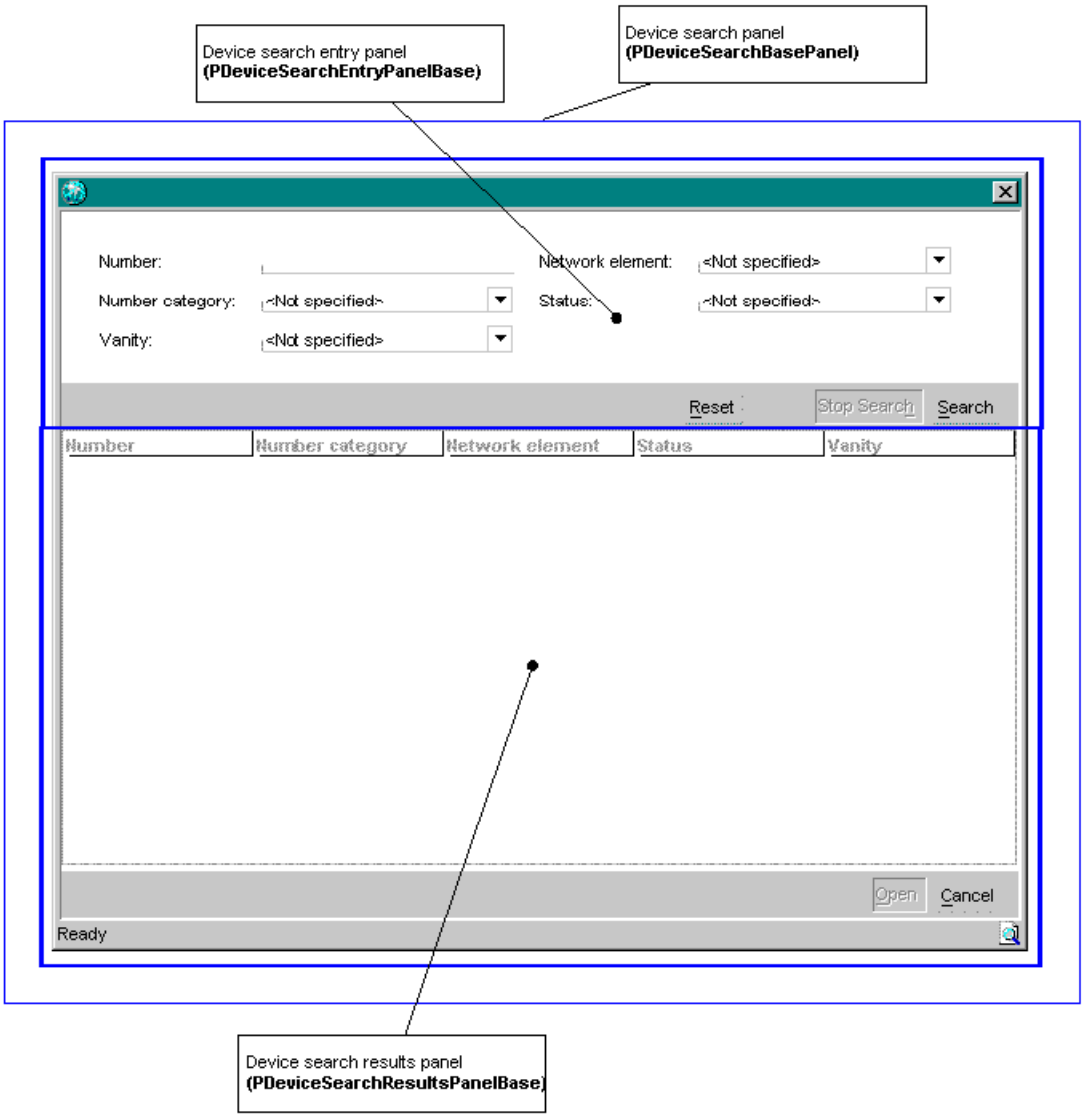
> **Note:** The sample template has prepopulation enabled.

Creating Custom Device Search Panels

You extend **PDeviceSearchDialog** to customize device search panels for a specific device.

The base device search panel appears as in Figure 17–3:

Figure 17–3 Base Device Search Panel



You can override most of these methods in your extended class to perform custom actions. For information about the public methods in this class, see **PDeviceSearchDialog** in the Customer Center *JavaDocs*.

PDeviceSearchDialog contains the panel **PDeviceSearchBasePanel** which contains a search entry panel and a search results panel.

- The search *entry panel* contains the fields that a search will be based on.
- The search *results pane*l contains a table that shows the results of a search.

PDeviceSearchBasePanel reads the names of the entry panel and results panel class names from the Configurator properties file **(WizardCustomizations.properties)**. You configure the properties file by using Customer Center SDK Configurator. See "Configuring Service and Device Panel Layouts by Using Configurator".

For information about the public methods in this class, see **PDeviceSearchBasePanel** in the Customer Center *JavaDocs*.

Creating Custom Device Search Entry Panels

You extend **PDeviceSearchEntryPanelBase** to customize device search entry panels for a specific device.

The base device search entry panel appears as follows:

You can override most of these methods in your extended class to perform custom actions. For information about the public methods in this class, see **PDeviceSearchEntryPanelBase** in the Customer Center *JavaDocs***.**

Sample Search Entry Subclass Template

Customer Center SDK includes the subclass template *ccsdk_home*/**CustomerCareSDK/CCExamples/TelcoFramework/DeviceSearchEntryPanelTemplate.txt**. Use this as a starting point to create a device search entry panel subclass from **PDeviceSearchEntryPanelBase**.

The sample entry panel uses the status and device ID fields as the search criteria. You can add more search fields to this file by using JBuilder.

Creating Custom Device Search Results Panels

You extend **PDeviceSearchResultsPanelBase** to customize device search results panels for a specific device.

You can override most of the public methods in your extended class to perform custom actions. For information about the public methods in this class, see **PDeviceSearchResultsPanelBase** in the Customer Center *JavaDocs***.**

Sample Search Results Subclass Template

Customer Center SDK includes the subclass template *ccsdk_home*/**CustomerCareSDK/CCExamples/TelcoFramework/DeviceSearchResultsPanelTemplate.txt.** Use this as a starting point to create a device search results panel subclass from **PDeviceSearchResultsPanelBase**.

The sample results panel has a **PIASpecSpreadSheet** table. This table contains a column for device status. You can rename this template and add more columns to this table.

Utility Class CCTelcoUtility

Customer Center SDK includes the utility class **CCTelcoUtility** that provides methods for implementing clients. Most of the methods implement cache information retrieved from the database so that subsequent calls do not require additional calls to the database.

> **Important:** If there are any changes on the server side, you must shut down and restart Customer Center to apply the changes.

For information about the public methods in this class, see **CCTelcoUtility** in the Customer Center *JavaDocs***.**

Configuring Service and Device Panel Layouts by Using Configurator

You use Customer Center SDK Configurator to configure service and device panel layout and to specify any custom panels you created.

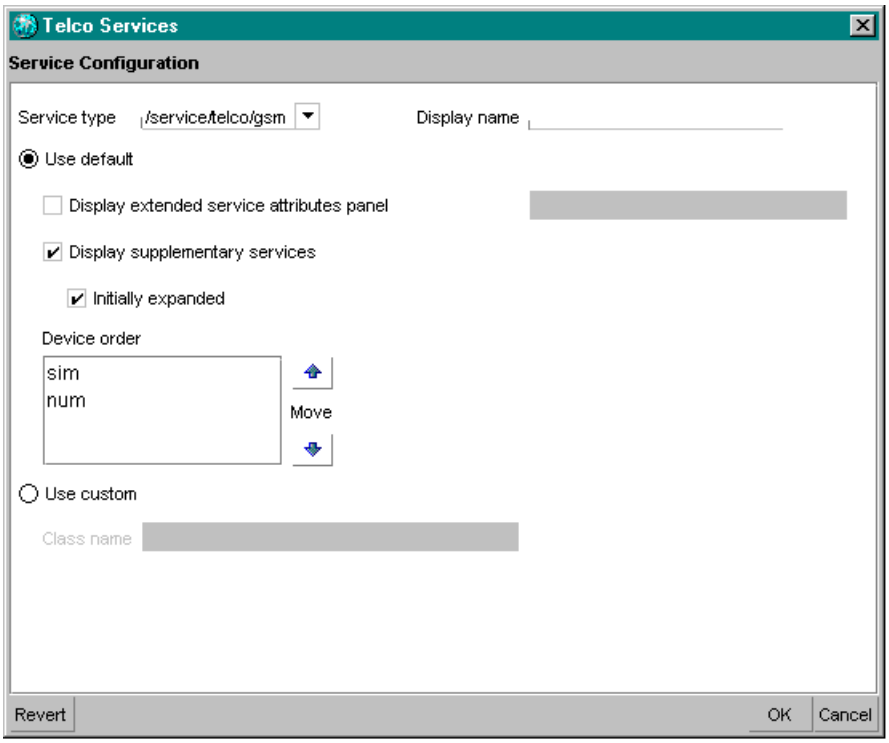
Telco Service Configurator

To configure the layout of the service panel for a telco service:

1. Choose **Tools - Telco - Service** from the Configurator main menu.

   The following screen in Figure 17–4 is displayed:

Figure 17–4 Telco Service Configurator



2. Configure the options in Table 17–2 as required:

Table 17–2 Telco Service Configurator Options

Option	Service Panel Action	Default
Service type	From the pull-down list, select a service type, such as **/service/telco/gsm**, for which the search panel configurations should apply.	**One of the available services**
Display name	Enter the text to be displayed for the service panel.	**Blank text box**
Use default	Select this option to use the default service panel.	**Selected**
Display extended service attributes panel	Select this option to display an extended service attributes panel.	**Do not display**

Table 17–2 (Cont.) Telco Service Configurator Options

Option	Service Panel Action	Default
Display extended service attributes panel	Enter the extended service attributes panel class name.	**Blank text box**
Display supplementary services	Select this option to display the supplementary services panel.	**Do not display**
Initially expanded	Select this option to display the supplementary services panel in expanded mode.	**Do not expand**
Device order	Arrange the listed devices in the order you want to display their panels. **Note:** Only devices relevant to the specified service type appear in the list.	**random**
Use custom	Select to use a custom service panel. **Note:** If you use a custom panel, the fields under the **Default** button are unavailable. You must configure the layout attributes of your panel within your subclass.	**Not selected**
Custom: Class name	Enter the custom service panel name.	**Blank text box**

When you save your Configurator session, customizations are written to the **WizardCustomizations.properties** properties file.

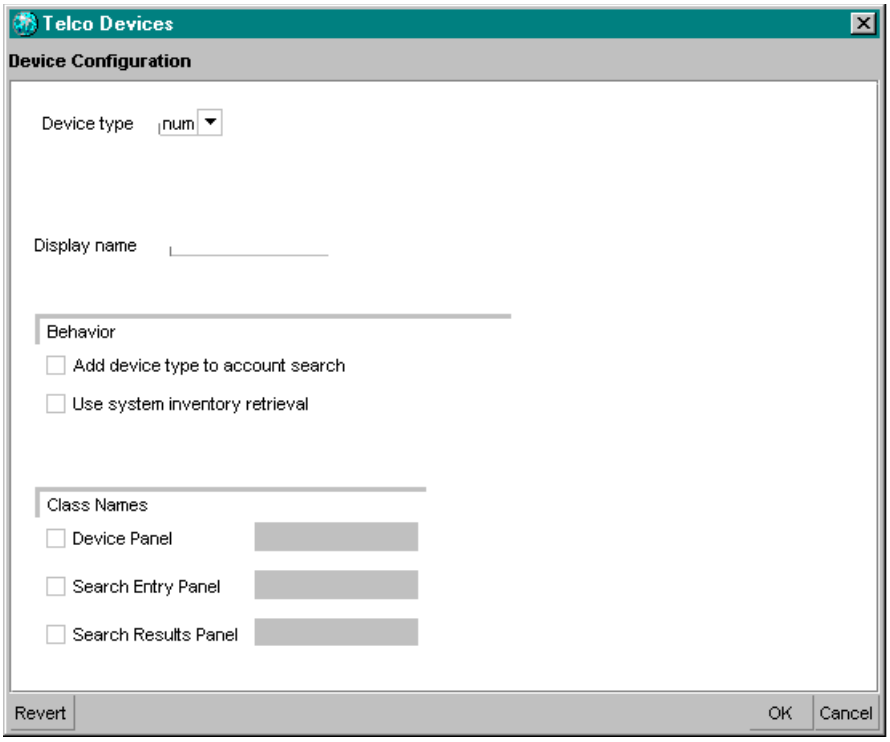
Telco Device Configurator

To configure the layout of device panels for a Telco service:

1. Choose **Tools - Telco - Device** from the Configurator main menu.

   The following screen as shown in Figure 17–5 is displayed:

Figure 17–5 Telco Device Configurator



2. Configure the options in Table 17–3 as required:

Table 17–3 Telco Device Configurator Options

Option	Device Panel Action	Default
Device type	Select the name of the device type for which properties are being defined.	**One of the available devices**
Display name	Enter the name that should be displayed in the device panel.	**Blank text box**
Add device type to account search	Select this option to allow CSRs to search for this device type.	**Do not allow search**
Use system inventory retrieval	Select this option to enable automatic search for the first available device in the system and populate the device panel with the information for that device. See "About Device Prepopulation".	**Do not use system inventory retrieval**
Device panel	Select this option and specify the class name for the device panel to be displayed.	**Blank text box**
Search entry panel	Select this option and specify the class name for the device search entry panel to be used for this device.	**Blank text box**
Search results panel	Select this option and enter the class name for the device search results panel to be used for this device.	**Blank text box**

When you save your Configurator session, customizations are written to the **WizardCustomizations.properties** properties file.

About Device Prepopulation

Prepopulation can speed up CSR account creation and device updating by preselecting a device from the BRM database for the CSR.

- **Prepopulation Enabled**

  If prepopulation is enabled, the next available device of the type is selected, and the device panel is populated with information specific to this device.

  By default, when prepopulation is enabled, the device panel base class searches the BRM database to retrieve the next available device. You can override this behavior with your own prepopulation logic. See "Creating Custom Device Panels".

- **Prepopulation Disabled**

  If prepopulation is disabled, the device panel is not populated with any information when the CSR selects a device type. The CSR must use the search dialog box to select a device.

18

About Managing Prepaid Services and Extended Rating Attributes

This chapter describes how to set up prepaid services, supplementary services, and extended rating attributes (ERAs) in your Oracle Communications Billing and Revenue Management (BRM) system.

For background information about wireless services, see "About Integrating Wireless Services".

About BRM Prepaid Services

A *BRM service* is the mechanism by which services are managed in BRM. They are identified by classes and objects, such as **/service/telco/gsm/telephony**.

Telco services are the services that are provisioned on a telco network. There are groups of telco services, such as telephony, data, and fax. For example, with GSM services, there are multiple varieties of GSM service in each GSM service group: the voice service group includes normal voice telephony service (T11) and emergency-only telephony service (T12).

A single service can be used to offer multiple types of a service. For example, the BRM **/service/telco/gsm/telephony** service is used for GSM normal voice service and emergency-only voice service. When you create products in Pricing Center, you use provisioning tags to assign different types of telco services to them. For more information, see "About Provisioning Tags for Telco Services".

The services in Table 18–1 are used to offer telco services:

Table 18–1 Services for Telco

Service	Description
/service/telco/gsm/telephony	Supports GSM voice services.
/service/telco/gsm/fax	Supports GSM fax services.
/service/telco/gsm/data	Supports GSM Internet access for corporate access or WAP access.
/service/telco/gsm/sms	Supports GSM Short Message Service (SMS).

Your price list can include deals for one or more services. For example, you might have a plan that includes just the GSM telephony service (**/service/telco/gsm/telephony**) and a plan that includes all BRM GSM services. You can create different usage and

subscription rates for different services. For more information, see "Setting Up GSM Wireless Pricing".

About Telco Service Logins and Passwords

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

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

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

  For example:

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

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

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

  To customize how IDs are generated, you customize the PCM_OP_CUST_POL_PREP_LOGIN policy opcode.
- The default password is **password**. You cannot change the password when a service is being added, but you can change it later in either Customer Center or Self-Care Manager.

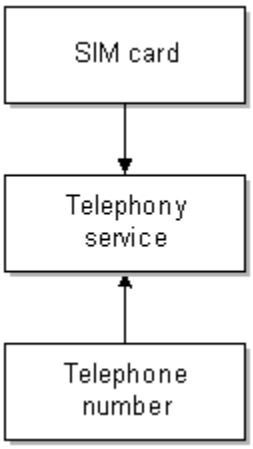
  To customize how passwords are generated, you customize the PCM_OP_CUST_POL_PREP_PASSWD policy opcode.

About Associating SIM Cards and Numbers with Services

Use Customer Center to associate SIM cards and telephone numbers with telco services and to change the SIM card and number for a customer's service.

SIM cards and telephone numbers are not associated with each other. Instead, you associate SIM cards and telephone numbers with the appropriate service, as shown in Figure 18–1:

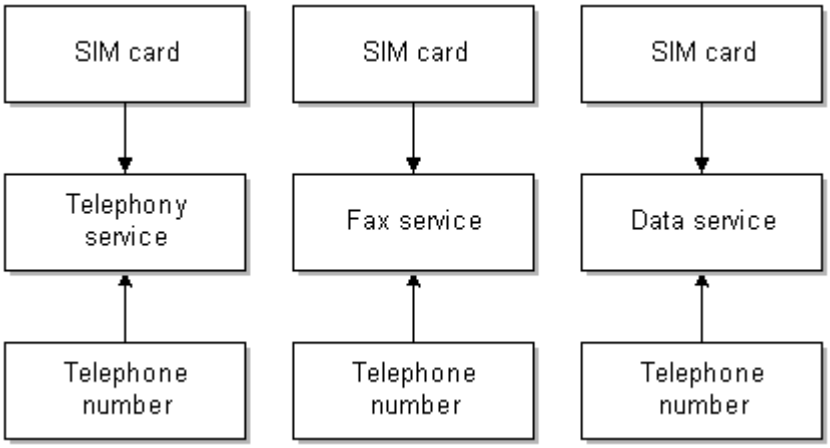
Figure 18–1 SIM Card to Service Association



You can associate SIM cards and numbers in a variety of ways. For example, a customer can own multiple services, SIM cards, and numbers.

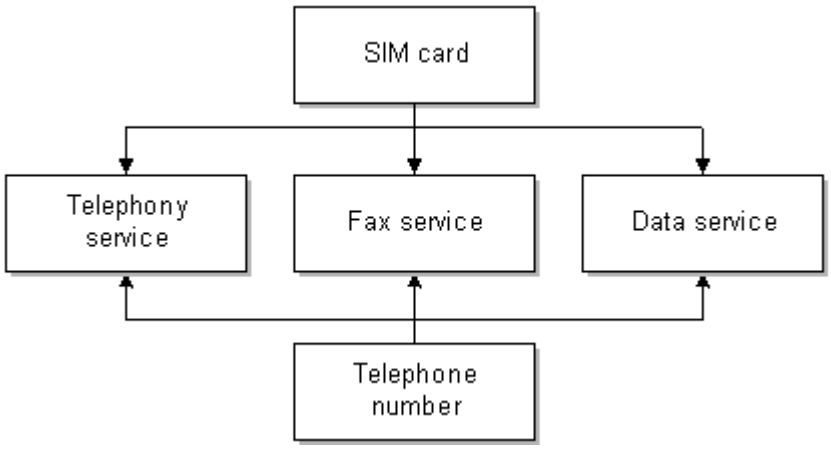
Figure 18–2 shows associations for three services, each with its own SIM card and number:

Figure 18–2 Three SIM Cards to Services Associations



You can also associate SIM cards and numbers with multiple services as shown in Figure 18–3:

Figure 18–3 SIM Card to Multiple Services Associations



You can customize the SIM card and number policy opcodes, PCM_OP_SIM_POL_DEVICE_ASSOCIATE and PCM_OP_NUM_POL_DEVICE_ASSOCIATE, to customize how services, SIM cards, and numbers can be associated. By default, the following business policies are used:

- Each account can have one or more SIM cards and one or more telephone numbers.
- You can use one SIM card and number for all services, or you can use multiple SIM cards and numbers; for example, you can use a separate SIM card and number for each service.
- A SIM card can be associated with multiple services only if the services are of different types, such as fax and telephony, and each service must belong to the same network element. For example, two **/service/telco/gsm/telephony** services can be associated with the same SIM card to support dual lines on one SIM card.
- A number can be associated with multiple services, with the following restrictions:
  - All services must belong to the same account.
  - All services must belong to the same network element.
  - Each service must be of a different type. For example, you cannot associate one number with two telco services. To do so, you need to customize the policy opcode for the device type, PCM_OP_SIM_POL_DEVICE_ASSOCIATE or PCM_OP_NUM_POL_DEVICE_ASSOCIATE.
- With the exception of the telephony service, a SIM card can share a service association with only one telephone number. For example, if you associate a SIM card with a fax service and a data service, the phone number must be the same for the fax service and the data service.

  If you associate a SIM card with two telephony services, you must associate a different number for each of the telephony services.
- A SIM card and number associated with the same service must have the same network element.
- A SIM card and number associated with the same service must belong to the same brand.

For more information, see "How Accounts Are Created and Managed".

About Assigning SIM Cards

When assigning SIM cards in Customer Center:

- You can manage SIM cards by brand.
- You cannot change the IMSI number, network element, or status.

For information about managing SIM cards in Customer Center, see information about GSM accounts in the Customer Center Help.

For information about managing your SIM card inventory, see "About Managing SIM Card Inventory".

About Assigning Numbers

When assigning numbers in Customer Center:

- You can manage numbers by brand.
- You cannot change the network element, number category, vanity flag, or status.
- If an account has more than one telephone number, you can specify one of them as the primary number. If Pipeline Manager processes an event that includes the IMSI, and if the IMSI is the same for multiple telephone numbers, Pipeline Manager uses the primary number to look up the service.

  Note: Customer Center does not support adding more than one number to an account, but you can create a custom application to do this.

- You can assign an IMEI number, but it is not required.
- You can assign vanity numbers to charge a different amount for specific numbers.

For information about managing your telephone number inventory, see "About Managing Telephone Numbers".

About Provisioning Tags for Telco Services

To implement supplementary services, service extensions such as bearer services, and extended rating attributes (ERAs) for telco services, you define provisioning tags. You then use Pricing Center to include provisioning tags in products. A tag becomes available to an individual account and service when a product containing the tag is purchased. This is also known as product-level provisioning.

Use one of these methods to define provisioning tags for telco services:

- The Provisioning Tags application in Pricing Center. You can create provisioning tags that include existing ERAs only.

  See "About the Provisioning Tags Application" and Provisioning Tags Help.

- The **pin_telco_tags**_*service* file (for example, **pin_telco_tags_gsm**) and the **load_pin_telco_tags** utility. Use this method to create provisioning tags with custom ERAs.

  See "Defining Provisioning Tags for Telco Services by Using the pin_telco_tags file".

Use these methods for provisioning tags for telco services, except for the following cases:

- For a provisioning tag that creates an ERA or other type of profile that you want to associate with a discount. Services Framework Manager does not process provisioning tags associated with discounts.
- For a provisioning tag that populates default values to a profile when creating it. Services Framework Manager cannot populate default values to the profiles.

In both of these cases, use the provisioning tag framework to create provisioning tags. See "Using the Provisioning Tag Framework" in *BRM Setting Up Pricing and Rating*.

Service-level provisioning tags are stored in service-specific **/config/telco** objects, such as **/config/telco/gsm/telephony**.

You can also use the **pin_telco_tags**_*service* file to create account-level ERAs, which do not depend on a specific service. These ERAs are stored in **/config/account_era** objects.

Important: You cannot use the Provisioning Tags application to add account-level ERAs to provisioning tags.

For more information, see:

- About GSM Supplementary Services
- About Extended Rating Attributes for Telco Services

Some provisioning tags are available by default when you install GSM Manager. For a list of these tags, see "Default Account-Level ERAs" and "Default Service-Level ERAs".

For information on creating provisioning tags for non-telco services, see "Working with Provisioning Tags" in *BRM Setting Up Pricing and Rating*.

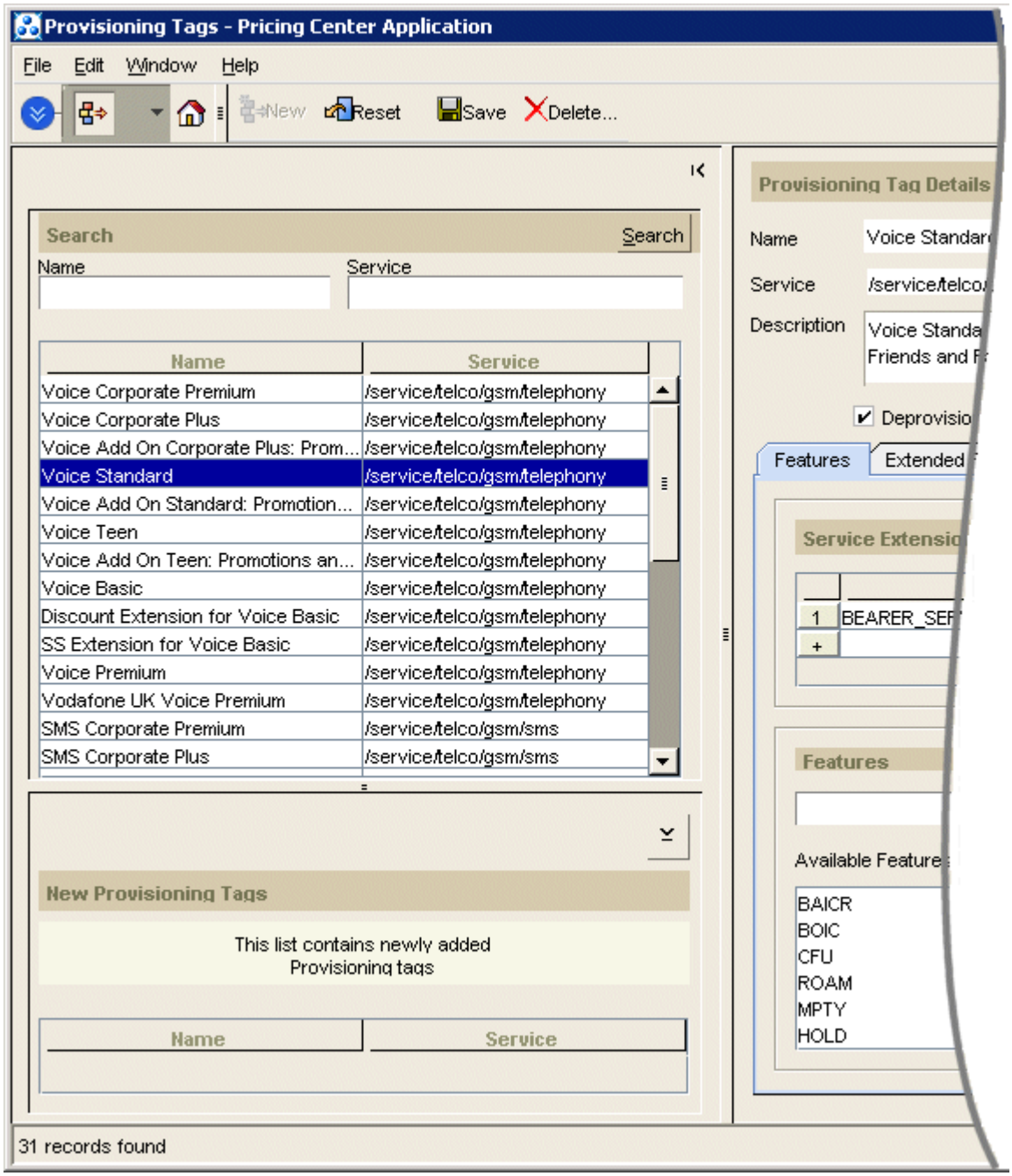
About the Provisioning Tags Application

You can create provisioning tags for telco services in the Provisioning Tags application, which is part of Pricing Center.

You can use this application to do the following:

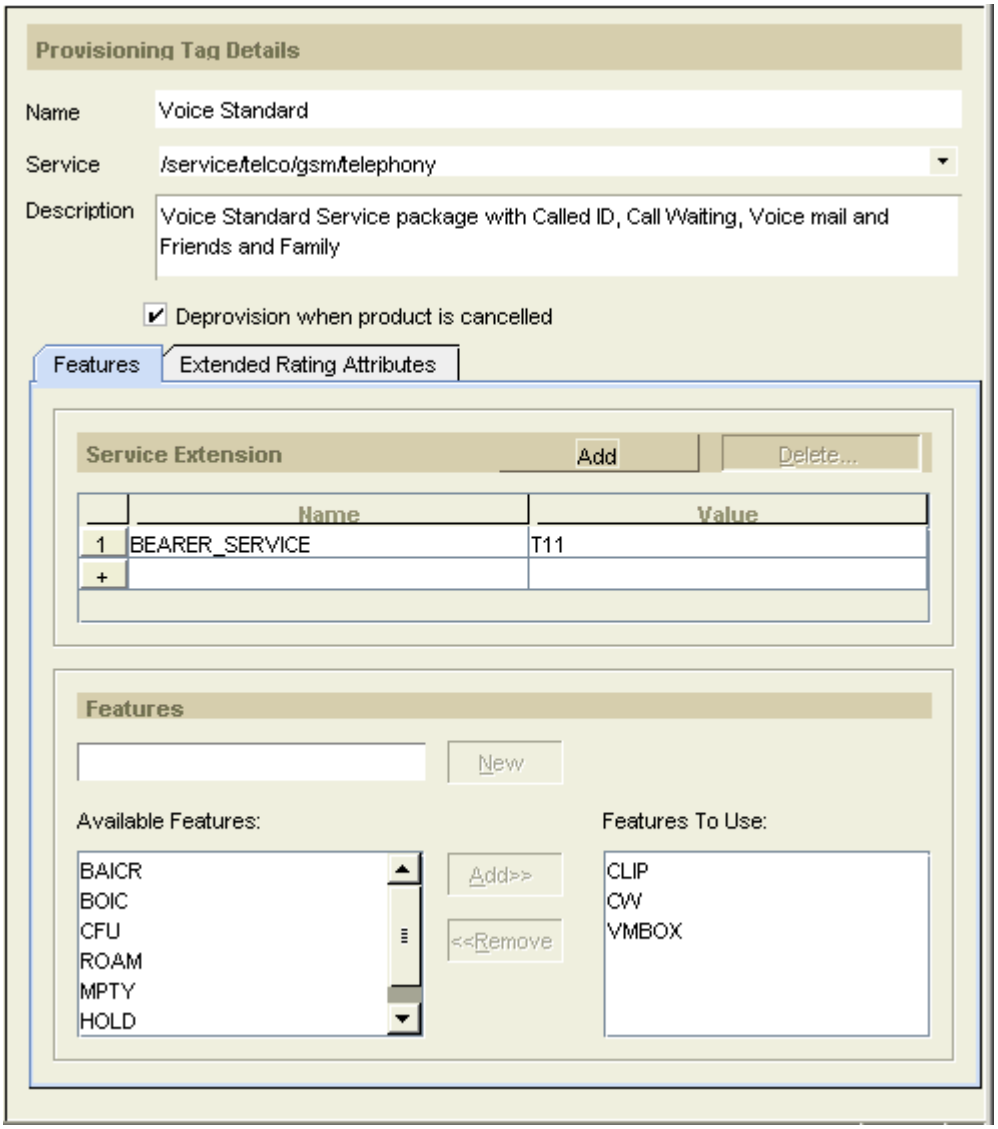
- Create new provisioning tags or search for, display, and edit existing provisioning tags for telco services as shown in Figure 18–4:

Figure 18–4 Provisioning Tags Application



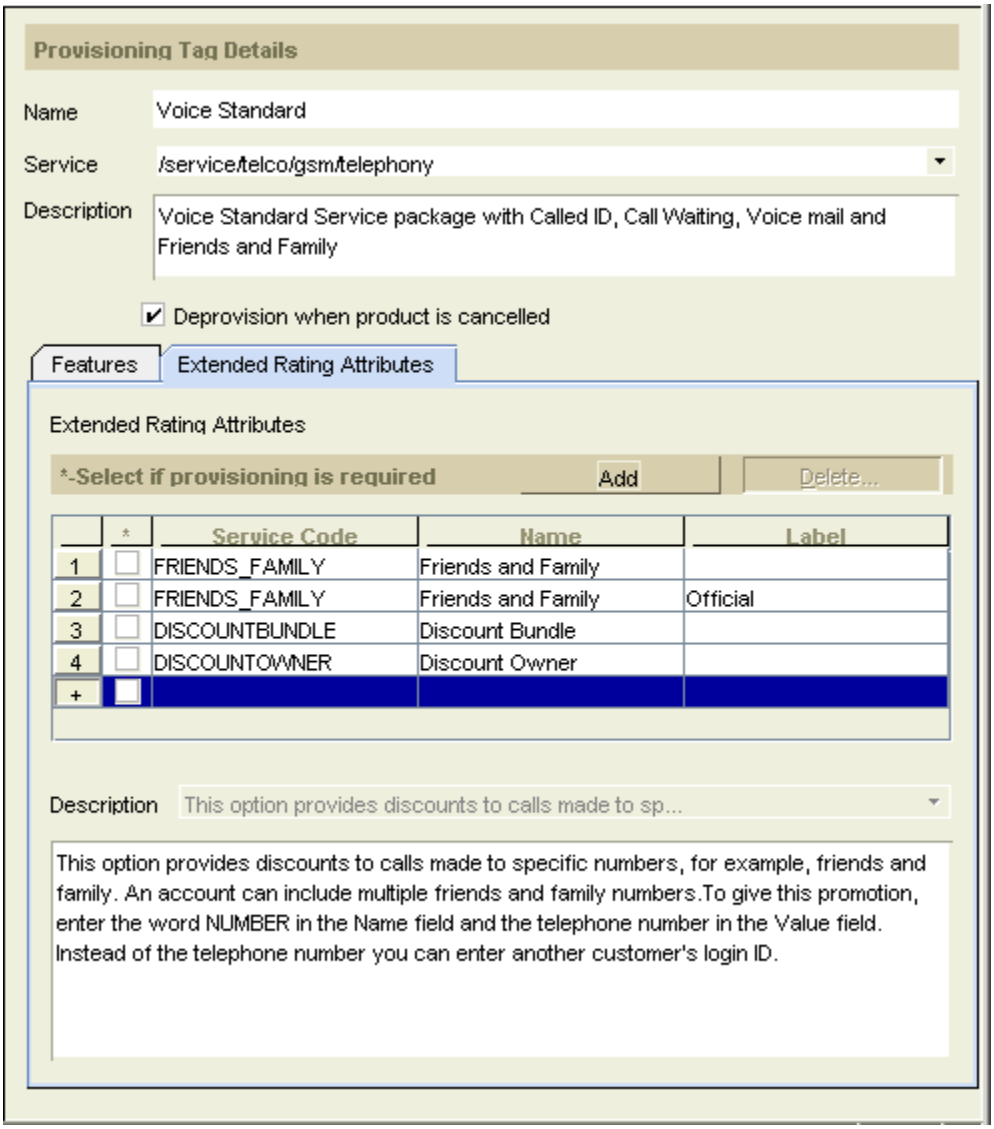
- Add or delete service extensions, such as bearer services, and features for new or existing provisioning tags as shown in Figure 18–5:

Figure 18–5 Service Extensions Window



- Add or delete ERAs and ERA labels for new or existing provisioning tags as shown in Figure 18–6. An ERA label is an individual ERA list. An ERA can have multiple lists:

Figure 18–6 ERA Window



You can include only ERAs that already exist in the BRM database, although you can add new label names. To create custom ERAs, see "Defining ERAs for Telco Services" and "Creating ERAs" in *BRM Setting Up Pricing and Rating*.

> **Note:** The ERA names and descriptions displayed in Provisioning Tags are from the **era_descr.***locale* file. See "Customizing ERA Names and Descriptions for Client Applications" in *BRM Setting Up Pricing and Rating*.

For more information, see Provisioning Tags Help.

Examples of Provisioning Tags for Prepaid Services

You can create different types of provisioning tags; for example:

- A provisioning tag for a single bearer service, such as a type of voice service.

- A provisioning tag for one or more supplementary services without a bearer service. A product with this type of provisioning tag is typically included in an add-on plan because the customer must have the service already before adding the supplementary services.
- A provisioning tag for one or more service-level ERAs. This type of provisioning tag can be used only in an add-on plan.
- A provisioning tag can include combinations of a bearer service, supplementary services, and service-level ERAs.

For example, you might include two different provisioning tags for a GSM telephony service product:

- The VoicePremium provisioning tag implements the following:
  - A voice bearer service
  - The Call Forwarding supplementary service
  - The Home Cell Assignment service ERA
- The VoiceFamily provisioning tag implements the following:
  - A voice bearer service
  - The Caller ID supplementary service
  - The Friends and Family service ERA

You might also create products such as these:

- A product that implements a voice bearer service.
- An add-on product that implements Call Forwarding and the Home Cell Assignment ERA.
- An add-on product that implements Caller ID and the Friends and Family ERA.

Important: You cannot directly change the status of supplementary services. Instead, you change the status by changing the status of the products that they were purchased with.

For example, to inactivate a Call Forwarding supplementary service, you inactivate its product. However, when you do so, you inactivate all other products, supplementary services, and service-level ERAs that were purchased with that product.

Therefore, you should create products that allow you to manage services after the products are purchased. For more information, see the Customer Center Help.

About GSM Supplementary Services

GSM supplementary services are features such as call forwarding and call blocking. They are not implemented as BRM services. Instead, they are implemented by using product-level provisioning.

For example, you might have a product in your price list that includes the GSM telephony service and a provisioning tag that implements the call forwarding supplementary service.

Note: Supplementary services can be used only with GSM services.

Value-added services, such as voice mail, are similar to supplementary services. The difference is that value-added services are not part of the GSM network standard.

Note: In Customer Center, and in this documentation, supplementary services and value-added services are referred to collectively as *supplementary services*.

You include supplementary services in products by using provisioning tags. You can create products that add supplementary services to an existing account; for example, a product that adds call forwarding.

You cannot activate supplementary services in Customer Center. After BRM provisions a supplementary service, a customer usually activates the supplementary service using the telephone keypad. For example, a customer can define a number to use for call forwarding.

For more information, see "About Provisioning Tags for Telco Services".

How Supplementary Services Are Stored in BRM

A customer's supplementary services are defined in the service objects owned by the customer's account. For more information, see **/service/telco/gsm**.

You define system-wide supplementary services in **/service/telco/gsm** objects. For example, supplementary voice telephony services are defined in **/service/telco/gsm/telephony**.

About Extended Rating Attributes for Telco Services

Extended rating attributes (ERAs) provide discounts and promotions based on a specific attribute, such as a telephone number. For example, you use ERAs to offer special friends and family rates or a birthday discount.

A friends and family ERA type can have multiple lists. Each individual list is identified as an ERA label. During rating, BRM can apply different rates based on the label.

For general information about ERAs, see "About Extended Rating Attributes".

BRM contains default ERAs for telco services. For lists of default ERAs, see "Default Account-Level ERAs" and "Default Service-Level ERAs". Default ERAs are loaded when you install a telco service manager, such as GSM Manager.

To create ERAs for telco services, see "Defining ERAs for Telco Services".

For more information, see:

- About Configuring ERAs for Individual Customers
- How ERAs for Telco Services Are Stored in BRM

About Configuring ERAs for Individual Customers

When a customer purchases a product that uses telco provisioning tags, the primary telco service is provisioned. You then use the Customer Center **Promotion** tab to configure the customer's ERAs. For more information, see the Customer Center Help.

Note:

- ERA codes are defined in Pipeline Manager configuration files. When you enter ERA data in Customer Center, the data you enter depends on how you configure the ERAs in Pipeline Manager. For example, you might enter a number in an ERA. That number is mapped to a value in ERA configuration.
- When configuring ERAs in Customer Center, use only uppercase letters, ASCII 7-bit punctuation, and no spaces.

You do not configure any primary services or supplementary services after an account is created. The customer usually performs this type of configuration using the handset. For example, a customer might define call forwarding numbers.

How ERAs for Telco Services Are Stored in BRM

A customer's ERA configurations are stored in profile objects:

- The **/profile/serv_extrating** object, which stores the service-level ERA configuration, is linked to the telco service objects owned by the customer's account.
- The **/profile/acct_extrating** object, which stores the account-level ERA configuration, is linked to the account object that owns the telco services.

You define system-wide definitions for service-level ERAs in **/config/telco/***service* objects. For example, supplementary voice telephony services are defined in **/config/telco/gsm/telephony**. Account-level ERAs are defined in **/config/account_era** objects. See "About Provisioning Tags for Telco Services".

Defining Provisioning Tags for Telco Services by Using the pin_telco_tags file

This section describes defining provisioning tags through the **pin_telco_tags** file for a specific telco service. For example, you use **pin_telco_tags_gsm** for GSM provisioning tags.

You can include service-level ERAs, supplementary services, and service extensions in a provisioning tag defined in a **pin_telco_tags** file.

You can use the Provisioning Tags application in Pricing Center instead of the **pin_telco_tags** file to define provisioning tags for telco services. But you cannot create custom ERAs using Provisioning Tags. For information, see "About the Provisioning Tags Application" and Provisioning Tags Help.

To define provisioning tags using the **pin_telco_tags** file:

- Configure provisioning tags in the **pin_telco_tags**_*service* file. See "Configuring Provisioning Tags in the pin_telco_tags File".
- Load the **pin_telco_tags**_*service* file into the BRM database with the **load_pin_telco_tags** utility. See "Loading the pin_telco_tags File".

You can also define account-level ERAs in the **pin_telco_tags** file. See "Defining Account-Level ERAs in the pin_telco_tags File"

Configuring Provisioning Tags in the pin_telco_tags File

To configure a provisioning tag in the **pin_telco_tags** file:

1. Open the **pin_telco_tags**_*service* file. For example, use **pin_telco_tags_gsm** for GSM services.

   The default **pin_telco_tags** files are in *BRM_Home*/**sys/data/config**. They include examples and instructions.

2. Use this syntax to add a provisioning tag:

```
provisioning_tag  "Class" "ProvTag" "PTDescription" "DelayedProvReqd"
service_extn      "Extension Type Name" "Extension Value"
features          "One Or More Feature Name String Values"
service_era       "ServiceERA" "StringIdERA" "StringIdServiceERADesc" "ProvBool" "ERALabel"
```

Enter each value in quotation marks.

A provisioning tag can be any combination of service extensions, features, and service-level ERAs. You do not need to include all three types of data in a tag.

Table 18–2 describes the provisioning tag syntax:

Table 18–2 Provisioning Tag Syntax

Tag Element	Value	Description
provisioning_tag	Class	The object that stores the tag. For example: **"/config/telco/gsm/telephony"**
provisioning_tag	ProvTag	The name of the provisioning tag. For example: **"DataPremium"**
provisioning_tag	PTDescription	The description of the provisioning tag. For example: **"Premium Data Service"**
provisioning_tag	DelayedProvReqd	Whether the tag is unprovisioned when the product containing the tag is canceled. The possible values are: ▪ **"y"** specifies that cancellation triggers unprovisioning. In most cases, use this setting. ▪ **"n"** specifies that cancellation does not trigger unprovisioning. Use this setting to leave a customer's service configuration unchanged. For example, you might want to leave a voice mailbox intact.
service_extn	Extension Type Name	The type of service extension. For example: **"BEARER_SERVICE"**
service_extn	Extension Value	The code for a GSM bearer service or other service extension. For example: **"B46"** Codes are defined in the GSM specification. You must use the exact code that the network requires.

Table 18–2 (Cont.) Provisioning Tag Syntax

Tag Element	Value	Description
features	One or More Feature Name String Values	The GSM supplementary services that are provisioned when this product is purchased. The services are entered as codes, in one line. For example: **"CLIP" "CW"** These codes are defined in the GSM specification. You must use the exact code that the network requires. For a list of codes, see "Supported Supplementary Services".
service_era	ServiceERA	The service ERA code. For example: **"FRIENDS_FAMILY"**
service_era	StringIdERA StringIdServiceERADesc	The IDs for the ERA name and description. For example: **"12" "13"** You define a localized name and description for these IDs in the **era_descr.***locale*file. These names and descriptions appear in Customer Center. See "Customizing ERA Names and Descriptions for Client Applications" in *BRM Setting Up Pricing and Rating*.
service_era	ProvBool	Whether or not provisioning is required. The possible values are: ■ **"y"** specifies that provisioning is required. ■ **"n"** specifies that provisioning is not required.
service_era	ERALabel	The name of a list within the ERA. An ERA can have one or more lists. For example: **"MYFRIENDS"** **Note:** You cannot localize the ERA label. You cannot have duplicate label names associated with the same ERA code.

This example shows a provisioning tag for a telephony product that includes a bearer service, call waiting and voice mailbox supplementary services, and friends and family service-level ERAs:

```
# Standard Telephony Package
provisioning_tag  "/config/telco/gsm/telephony"  "Voice Standard" "Voice Standard
Service package with Called ID, Call Waiting, Voice mail and Friends and Family"
"y"
service_extn      "BEARER_SERVICE" "T11"
features          "CLIP" "CW" "VMBOX"
service_era       "FRIENDS_FAMILY" 12 13 "n" "MYFRIENDS"
service_era       "FRIENDS_FAMILY" 12 13 "n" "MYFAMILY"
```

Loading the pin_telco_tags File

Run the **load_pin_telco_tags** utility to load the contents of the **pin_telco_tags_***service* file: for example, the **pin_telco_tags_gsm** file: into the BRM database. This utility creates or updates **/config/telco/***service* and **/config/account_era** objects.

Caution: By default, the **load_pin_telco_tags** utility appends telco provisioning tags and account-level ERAs to the BRM database. But if you use the **-x** parameter, this utility overwrites existing telco provisioning tags and account-level ERAs. Do not use the **-x** parameter unless you are certain you want to overwrite existing objects.

Important: The **load_pin_telco_tags** utility requires a configuration file. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Note: You cannot create ERAs for individual brands. All ERAs can be used by any brand.

1. Edit the **pin_telco_tags**_*service* file to add the custom account-level ERAs. The default **pin_telco_tags_gsm** file in *BRM_Home***/sys/data/config** includes examples and instructions.
2. Save the **pin_telco_tags**_*service* file.
3. Use the following command to run the **load_pin_telco_tags** utility:

   ```
   load_pin_telco_tags pin_telco_tags_service
   ```

   For the complete command syntax, see "load_pin_telco_tags".
4. Restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
5. Restart Pricing Center.

To verify that the account ERAs were loaded, you can display the **/config** objects by using the Object Browser or use the **robj** command with the **testnap** utility. (See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.)

Defining ERAs for Telco Services

You can create both service-level and account-level ERAs. This section describes creating service-level ERAs. For more information about account-level ERAs, see "Defining Account-Level ERAs in the pin_telco_tags File".

To create and implement an ERA for a telco service:

1. Define the ERA in a provisioning tag.

   See "Defining Provisioning Tags for Telco Services by Using the pin_telco_tags file".

   Note: To use an ERA already defined in a default provisioning tag, you do not need to define a new provisioning tag, but you must perform the remaining steps.

2. Define how the ERA is rated. For example:

- To rate an ERA based on usage type, create a usage type corresponding to the ERA and customize the IRL_UsageType iRule. See information about defining usage types in Pricing Center Help and "Configuring the IRL_UsageType iRule for ERAs" in *BRM Setting Up Pricing and Rating*.

  The usage type code must match the value in the **IRL_UsageType.data** file for this type of ERA.

- To rate an ERA based on the label name, define a price or discount model selector for pipeline rating or a rate plan selector for real-time rating. The rate or discount is based on the label name. See "Using ERAs with Multiple Lists" in *BRM Setting Up Pricing and Rating*.

3. Include the provisioning tag when creating or modifying a product based on a telco service or on an account. See information about defining product purchase information in Pricing Center Help.

   Note: Although it is possible to include provisioning tags in discounts, BRM does not make use of them for telco services.

4. Add the names and descriptions to the **era_descr.***locale* file. Names and descriptions from this file are displayed in Customer Center and Provisioning Tags. See "Customizing ERA Names and Descriptions for Client Applications" in *BRM Setting Up Pricing and Rating*.

5. Configure the ERA for a specific customer in Customer Center. ERAs are called *promotions* in Customer Center. ERAs you configure are stored as profiles. For more information, see "About Configuring ERAs for Individual Customers" and the Customer Center Help.

   Note: For ERAs based on usage types, CSRs must enter values in Customer Center that exactly match the way values are specified in the **ISC_UsageType.isc** file. See "Configuring the IRL_UsageType iRule for ERAs" in *BRM Setting Up Pricing and Rating*.

6. (Optional) Create a profile sharing group to share the ERA with other accounts. See information about creating a profile sharing group in the Customer Center Help.

   Note:

   - Even though an account is qualified to use ERAs, you do not have to implement them in the account.
   - Service-level ERA profile information is not automatically transferred between plans during plan transition or generation change. If the two plans have some common provisioning tags, the ERA profile information can be reconfigured in the new plan. See information about configuring service-level promotions in the Customer Center Help.
   - BRM does not validate any data entered when configuring ERAs; for example, telephone numbers for the friends and family discount. To create validation rules for these entries, edit the PCM_OP_CUST_POL_VALID_PROFILE policy opcode.

Defining Account-Level ERAs in the pin_telco_tags File

Account-level ERAs apply to any activity in an account no matter which service is involved.

You define account-level ERAs in any **pin_telco_tags**_*service* file. You can use the define account-level ERAs in the same file you used for telco provisioning tags.

> **Note:** You cannot use the Provisioning Tags application for account-level ERAs.

To define an account-level ERA:

1. Open a **pin_telco_tags**_*service* file. You can define an account-level ERA in any **pin_telco_tags** file, because they are not associated with a particular service.

   The default **pin_telco_tags** files are in *BRM_Home*/**sys/data/config**. They include examples and instructions.

   Account-level ERAs are added in a block at the end of the file.

2. Use this syntax to add an account-level ERA:

```
account_era  "AccountERA" "StringIdERA" "StringIdAccountERADesc"
```

Table 18–3 describes the account ERA syntax:

Table 18–3 Account ERA Syntax

Value	Description
AccountERA	The account ERA code. For example: **"SPECIAL_DAY"**
StringIdERA StringIdAccountERADesc	The IDs for the ERA name and description. For example: **"2" "3"** You define a localized name and description for these IDs in the **era_descr.***locale*file. These names and descriptions appear in Customer Center. See "Customizing ERA Names and Descriptions for Client Applications" in *BRM Setting Up Pricing and Rating*.

For example:

```
account_era "SPECIAL_DAY" "2" "3"
```

Supported Supplementary Services

Table 18–4 shows the supplementary services supported by GSM Manager. You select from these codes when configuring GSM provisioning tags in Pricing Center.

These codes are defined in the GSM provisioning DTD file (**GSM.dtd**), which is used by the GSM provisioning components. If you customize the GSM provisioning components to add supplementary services, you must be sure to use the new codes in the provisioning tags.

Table 18–4 GSM Manager Supported Supplementary Services

Supplementary Service	Code
Advice of charge (charging)	AOCC
Advice of charge (information)	AOCI
Barring all incoming calls	BAIC
Barring incoming calls when roaming outside the home PLMN country	BAICR
Barring all outgoing calls	BAOC
Barring all incoming calls when roaming outside the home PLMN country	BICRO
Barring all outgoing international calls	BOIC
Barring all outgoing international calls except those directed to the HOME PLMN country	BOICXH
Completion of calls to busy subscribers	CCBS
Call deflection	CD
Call forwarding on mobile subscriber busy	CFB
Call forwarding on mobile subscriber not reachable	CFNRC
Call forwarding on mobile subscriber no reply	CFNRY
Call forwarding unconditional	CFU
Calling line identification presentation	CLIP
Calling line identification restriction	CLIR
Name identification	CNAP
Connected line identification presentation	COLP
Connected line identification restriction	COLR
Call waiting	CW
Explicit call transfer	ECT
Call holding	HOLD
Multicall	MC
Enhanced multilevel precedence	MLPP
Multiparty	MPTY
Support of private numbering plan	SPNP
User-to-user signaling	UUS
Voice/fax mail service	VMBOX

Default Account-Level ERAs

Table 18–5 summarizes the default account-level ERAs:

Table 18–5 Default Account-Level ERAs

Function	Name	Pipeline Manager Name
Assign an account to a business segment for business intelligence reporting.	Business Intelligence Segment ERA See information about adding a business intelligence segment promotion in the Customer Center Help.	DATAWAREHOUSE
Assign an account to a customer group, or assign an arbitrary quality level to an account.	Customer type or quality ERA See information about adding a customer type or quality promotion in the Customer Center Help.	CLASSIFICATION
Assign an account to a group of accounts that share certain telephony properties.	Corporate agreement ERA See information about adding a corporate agreement promotion in the Customer Center Help.	CORPORATE
Create a closed user group, such as a group of all mobile numbers in a company.	Closed user group ERA See information about adding a closed user group promotion in the Customer Center Help.	CLOSEDUSERGROUP
Give a discount for calls made on a specific calendar date.	Special day discount ERA See information about adding a special day discount promotion in the Customer Center Help.	SPECIAL_DAY
Use a pipeline rate plan.	Pipeline account-level rate plan ERA See information about adding a pipeline account-level rate plan promotion in the Customer Center Help.	RATEPLAN

Default Service-Level ERAs

Table 18–6 summarizes the default service-level ERAs:

Table 18–6 Default Service-Level ERAs

Function	Name	Pipeline Manager Name
Specify a discount account that is used for calculating volume discounts for multiple accounts. **Note:** You can calculate volume discounts for multiple accounts by creating a discount. Discounts and the DISCOUNTACCOUNT ERA are mutually exclusive. This ERA is supported for batch rating only.	Hierarchical discount account ERA See information about adding a hierarchical discount account promotion in the Customer Center Help.	DISCOUNTACCOUNT

Table 18–6 (Cont.) Default Service-Level ERAs

Function	Name	Pipeline Manager Name
Assign a Quality of Service to a service.	Service-level agreement ERA See information about adding a service-level agreement promotion in the Customer Center Help.	SERVICELEVEL
Define the home cells for a customer and allow discounts while calling from this area.	Home cell assignment ERA See information about adding a home cell assignment promotion in the Customer Center Help.	HOME_CELL
Provide cross-product volume discounts, as defined in pipeline discounts. **Note:** You can provide cross-product volume discounts by creating a discount. Discounts and the DISCOUNTACCOUNT ERA are mutually exclusive. This ERA is supported for batch rating only.	Pipeline discount model ERA See information about adding a pipeline discount model promotion in the Customer Center Help.	DISCOUNTMODEL
Provide discounts to calls made to specific numbers or regions, such as all numbers in a country or area code.	Home region code ERA See information about adding a home region code promotion in the Customer Center Help.	HOME_REGION
Provide discounts to calls made to specific numbers, such as those for friends and family members.	Friends and family ERA See information about adding a friends and family promotion in the Customer Center Help.	FRIENDS_FAMILY
Use a pipeline rate plan.	Pipeline service-level rate plan ERA See information about adding a pipeline service-level rate plan promotion in the Customer Center Help.	RATEPLAN

19

About Provisioning Services

This chapter describes how Oracle Communications Billing and Revenue Management (BRM) Services Framework provisioning works.

Before reading this document, see "Understanding the Services Framework".

About Services Framework Provisioning

Services Framework provisioning notifies external provisioning agents when an account or service changes. For example, it notifies an external provisioning agent when a customer adds or removes a service; adds, cancels, or replaces a device; or modifies a profile. After the external provisioning agent updates its records, Services Framework provisioning updates the account or service information in the BRM database.

For example, when a customer switches to a new wireless phone device, the following occurs:

1. Services Framework provisioning notifies the wireless phone carrier that the account is discontinuing its old device and switching to a new device.
2. The wireless phone carrier updates its records.
3. The wireless phone carrier responds to Services Framework provisioning that the records were successfully updated.
4. Services Framework provisioning updates the account by activating the new device and canceling the old device.

About Provisioning Telco and Non-Telco Services

Services Framework can provision both telco services and non-telco services.

- All telco service types (**/service/telco/*** objects) are automatically recognized and provisioned by Services Framework.
- Non-telco service types are provisioned only if they are listed in the **/config/service_framework/permitted_service_types** object. You specify the non-telco service types that Services Framework supports by using the **load_pin_service_framework_permitted_service_types** utility. See "Specifying the Non-Telco Services Supported by Services Framework".

About Service Orders

Services Framework provisioning uses *service orders* to alert external provisioning agents about account and service changes. The service order includes details about what changed, including the following:

- POID of the device, service, or profile object that changed
- The current status of the service order. See "About Service Order Status".
- The provisioning action to perform: activation, deactivation, suspension, reactivation, change, or ignore
- Specified fields from the **/service** or **/device** object. See "About Adding Details to the Service Order".

The service order is sent to the external provisioning agent and stored in the BRM database in an **/event/provisioning/service_order/telco/***service_name* object.

About Creating Service Orders for Supplementary Services

Supplementary services are added or removed from service objects (**/service/telco/***service_name*) when products and deals are purchased or canceled. Services Framework service management initially sets the service status to NEW. If supplementary services are not included with the service, Services Framework provisioning adds them to the service order.

Note: It is possible that some features are part of two or more products that are purchased separately. In this case, the one added later takes the status of the existing one and is not included in the service order.

Tip: Customer Center displays the supplementary services status in the **Service** tab.

You can localize the status flags for the service and the status for the supplementary service by using the **load_localized_strings** utility, which updates the **/strings** storable class based on localized configuration file. See "Creating a Localized Version of BRM" in *BRM Developer's Guide*.

About Creating Service Orders for Devices

Device service order creation is controlled by the device status stored in the configuration object (**/config/device_state**). If the device status is listed in the configuration object, a service order is created. The action associated with this service order is the one specified in the configuration for this device status entry.

The **/event/device/associate** and **/event/device/disassociate** events occur during an update services action. These events get device information needed to create the service order from the device objects. Only events for device types listed in the **/config/telco/provisioning** object are processed. The fields read from the device objects are specified in the **/config/telco/provisioning** object for this type of device.

When a device is changed, the **/event/device/state** event is generated as part of the device state change. PCM_OP_TCF_CREATE_SVC_ORDER reads the configuration object (**/config/telco/provisioning)** to determine if the device state requires that a

service order be created. If so, this opcode creates it as part of the service order event (**/event/provisioning/service_order/telco/***service_name)*.

The PCM_OP_TCF_PROV_POL_CREATE_SVC_ORDER policy opcode is called as part of the state transition and can be customized to change some value in the service order. This opcode calls the opcodes that update the service order.

About Creating Service Orders for Profile Changes

Profiles are added and removed from BRM service objects through the purchase and cancellation of products and deals. Services Framework provisioning includes profiles in service orders as directed by the provisioning configuration object **/config/provisioning/telco**. The PCM_OP_TCF_CREATE_SVC_ORDER opcode determines whether to create a service order based on data that has been added, changed, or removed from this object.

This opcode subscribes to the **/event/notification/profile/pre-modify** and **/event/notification/profile/modify** events generated by the PCM_OP_CUST_MODIFY_PROFILE opcode to create service orders for profile changes. Based on the value in the status field of the profile object, a service order is created and stored in the **/event/provisioning/service_order/telco/***service_name* event by capturing changes made to the profile object.

About Service Order Status

Like services, service orders have different statuses in their lifetimes. Table 19–1 shows the default values for service order status:

Table 19–1 Service Order Status Default Values

Service Order Status	Description
NEW	This is the initial service order state.
READY	A READY service order is ready to be sent to the provisioning agent. The provisioning functional modules have all the necessary data to completely fill in the service order and send it to the network provisioning interface.
PROCESSING	After the service order is received by the network provisioning interface, the state is changed to PROCESSING.
COMPLETED	If the service order is successfully processed and the devices are provisioned, the service order state changes to COMPLETED.
FAILED	If there are errors during device provisioning or in the network, the status of the service order is set to FAILED.

To view the status of a service order, use Event Browser.

About the Provisioning Process

BRM uses event notification to alert Services Framework provisioning that one of the following occurred:

- A service was created or modified.
- A device was associated with or disassociated from a service.
- A device was replaced with a new device.

- A device's state changed.
- A profile was modified.

> **Note:** To configure event notification to alert Services Framework provisioning that other events occurred, see "Setting Up Event Notification for Provisioning".

Services Framework provisioning then performs the following main functions:

- Generates a service order and adds specified details to it. See "About Adding Details to the Service Order".
- Retrieves the service order state transitions for the specified service or device type. See "About the Allowable Service Order State Transitions".
- Optionally tests the provisioning process by using the Network Simulator. See "Testing Provisioning Using BRM Network Simulator" for more information.
- Publishes the service order and, depending on the provisioning mode, either finishes processing or waits for a response from the network provisioning agent. See "About Provisioning Modes".

About Adding Details to the Service Order

Service orders contain information about the service, device, or profile that changed as well as the provisioning action to perform. You can also have other details about the service, device, or profile added to the service order before it is sent to the external provisioning agent. For example, you can add the quality of service (QoS) values and APN names to GSM service orders.

You specify the service, device, or profile object fields to add to the service order in the *BRM_Home*/**sys/data/config/pin_telco_provisioning** configuration file. You then load the file into the *provisioning configuration object* (**/config/telco/provisioning** and **/config/telco/provisioning/fieldlist**) by using the **load_pin_telco_provisioning** utility.

During the provisioning process, Services Framework determines the object fields to include in the service order by reading the provisioning configuration object:

- For telco services, the provisioning configuration object is **/config/telco/provisioning/***ServiceType*, where *ServiceType* is the service type passed in the input flist. For example, if the service type is **/service/telco/gprs/telephony**, the provisioning configuration object is **/config/telco/provisioning/gprs/telephony**.
- For non-telco services, the **/config/service_framework/permitted_service_types** object lists the provisioning configuration object to use.

To configure the service, device, and profile object fields to add to the service order, see "Specifying the Details to Add to the Service Order".

About the Allowable Service Order State Transitions

Service orders can be set to a NEW, READY, PROCESSING, COMPLETED, or FAILED state. When a service order is first created, it is set to the NEW state by default. The service order can then transition from the NEW state to a list of permitted states that you specify.

For each service order state, you define the valid states to which it can transition. For example, you specify whether service orders can transition from a NEW state to only a READY state or from a NEW state to either a READY state or a PROCESSING state.

You specify the allowable state transitions on a service-by-service basis in the *BRM_Home*/**sys/data/pin_telco_service_order_state** configuration file. You then load the file into the **/config/telco/service_order_state/*** database object by using the **load_pin_telco_service_order_state** utility.

Note: You can also configure BRM to call an opcode when a service order transitions from one state to another.

To configure the state transitions that are allowed, see "Specifying the Available States for Each Service Order".

About Provisioning Modes

After service orders are published, the provisioning process cannot continue until the network provisioning agent returns a response. You can configure whether Services Framework provisioning sends the service order directly to the agent and waits for a response or publishes the service order to a queue by setting the provisioning mode:

- **Queued provisioning mode**. Services Framework provisions service orders in two separate transactions. In the first transaction, Services Framework generates a service order and queues it for the external provisioning agent. In the second transaction, the external provisioning agent responds that provisioning failed or was successful and then Services Framework updates the service in the BRM database. This is the default provisioning mode.
- **Confirmed provisioning mode**. Services Framework publishes the service order immediately to the external provisioning agent, waits for a response, and updates the service in the BRM database in one transaction. Specifically, Services Framework:
  - Processes the request and converts the service order information into a provisioning payload object in XML format.
  - Sends the service order to the network provisioning agent through a TCP/IP connection.
  - Waits for a response from the network provisioning agent.

    If the wait exceeds the timeout value, the transaction is rolled back. See "Setting a Timeout Value for Requests Sent in Confirmed Mode".

You specify the provisioning mode on a service-by-service basis by using the **pin_service_framework_permitted_service_types.xml** configuration file. You then load the XML file into the BRM database's **/config/service_framework/permitted_service_types** object by using the **load_pin_service_framework_permitted_service_types** utility. See "Specifying the Non-Telco Services Supported by Services Framework".

You can also customize the PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER policy opcode to set an event's provisioning mode based on service order attributes. The policy opcode overrides the provisioning mode set in the **/config/service_framework/permitted_service_types** object.

Note: The **/config/service_framework/permitted_service_types** object and the PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER policy opcode are used primarily to configure *non-telco* service types. However, they can be used to specify the provisioning mode for telco service types.

Provisioning Process Opcode Flow

Services Framework provisioning generates service orders as follows:

1. A customer account creates or modifies a service, device, or profile. This generates a notification event.
2. The event notification system calls the opcode specified in the **/config/notify** object. By default, the PCM_OP_TCF_PROV_CREATE_SVC_ORDER opcode is called. See "Using Event Notification" in *BRM Developer's Guide*.
3. PCM_OP_TCF_PROV_CREATE_SVC_ORDER performs the following:
   a. Determines whether the service type passed in the input flist is supported by Services Framework. All telco service types and all service types listed in the **/config/service_framework/permitted_service_types** object are supported. See "About Provisioning Telco and Non-Telco Services".
   b. Determines the *provisioning configuration object* to use. See "About Adding Details to the Service Order".
   c. Determines the *service order configuration object* to use. See "Specifying the Available States for Each Service Order".
   d. Generates the service order business event (**/event/provisioning/service_order/telco/***).

   Note: The service order business event contains "telco" in its name for both telco and non-telco service types because the common substruct for holding the service order data is at the **/event/provisioning/service_order/telco** level.
4. The event notification system calls the opcode specified in the **/config/notify** object. By default, PCM_OP_TCF_PROV_HANDLE_SVC_ORDER is called.
5. PCM_OP_TCF_PROV_HANDLE_SVC_ORDER performs the following:
   a. Determines whether the service order status is NEW. If it is, the opcode calls PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE to update the service order status to READY.
   b. Determines whether to call Network Simulator by reading the **simulate_agent** entry in the CM **pin.conf** file. If **simulate_agent** is set to **1**, the opcode calls the PCM_OP_TCF_PROV_SIMULATE_AGENT opcode to simulate the provisioning flow with the CM. For information about Network Simulator, see "Testing Provisioning Using BRM Network Simulator".
   c. Determines the provisioning mode for the service type by reading the **/config/service_framework/permitted_service_types** object. The default is **Queued**. See "About Provisioning Modes".
   d. Calls the PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER policy opcode with the service order event details and provisioning mode.

6. The PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER policy opcode performs any custom actions and then returns to the calling opcode. By default, this policy opcode does nothing, but you can customize it to override the provisioning mode and modify service order event details.
7. PCM_OP_TCF_PROV_HANDLE_SVC_ORDER calls the PCM_OP_PROV_PUBLISH_SVC_ORDER opcode to publish the service order.
8. PCM_OP_PROV_PUBLISH_SVC_ORDER publishes the service order to **dm_prov_telco**.
9. **dm_prov_telco** determines the provisioning mode by reading the PIN_FLD_MODE flist entry. **dm_prov_telco** operates in Queued mode when the entry is **0** and Confirmed mode when the entry is **1**.
   - In Queued mode, **dm_prov_telco** queues the request in-memory and calls PCM_OP_TCF_PROV_SERVICE_ORDER_SET_STATE to update the service order status to PROVISIONING. See "About Provisioning Modes".
   - In Confirmed mode, **dm_prov_telco** sends the service order to the network provisioning agent through a TCP/IP connection and waits for a response. See "About Provisioning Modes".

After the network provisioning agent returns a response in flist format, Services Framework provisioning performs the following:

1. **dm_prov_telco** sends the response to the PCM_OP_PROV_PUBLISH_SVC_ORDER opcode.
2. PCM_OP_PROV_PUBLISH_SVC_ORDER passes the response to the PCM_OP_TCF_PROV_HANDLE_SVC_ORDER opcode.
3. PCM_OP_TCF_PROV_HANDLE_SVC_ORDER calls the PCM_OP_PROV_UPDATE_SVC_ORDER opcode to update the service order status.
4. PCM_OP_PROV_UPDATE_SVC_ORDER changes the service order status to **Processed** and generates the **/event/provisioning/service_order/telco/*** business event.
5. The event notification system calls the opcode specified in the **/config/notify** object. By default, the PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT opcode is called.
6. PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT updates the service's status and supplementary features.

Setting Up Services Framework for Provisioning

To set up Services Framework for provisioning:

- Specify the notification events that trigger provisioning. See "Setting Up Event Notification for Provisioning".
- Specify the details to add to the service order. See "Specifying the Details to Add to the Service Order".
- Specify the available state transitions for each service type. See "Specifying the Available States for Each Service Order".
- Configure service status change for device-to-service associations. See "Configuring Service Status Change for Device-to-Service Associations".

Setting Up Event Notification for Provisioning

BRM uses event notification to start the Services Framework provisioning process. You specify which notification events trigger provisioning by editing the event notification configuration file and then loading it into the database with the **load_pin_notify** utility.

To configure event notification for provisioning:

1. Open the *BRM_Home***/sys/data/config/pin_notify_telco** file in a text editor.
2. If your system has multiple configuration files for event notification, merge them with the **pin_notify_telco** file.
3. Add the following entry for each service or device type that you want to provision:

   ```
   OpcodeNumber    Flag    Event
   ```

   where:

   - *OpcodeNumber* specifies the hard-coded number for an opcode. To find an opcode's number, see the opcode header files (***.h**) in the *BRM_Home***/include/ops** directory.
   - *Flag* is the name of the flag to pass to the opcode when it is called by the event notification feature. **0** means no flag is passed.
   - *Event* is the name of the event that triggers the opcode. You can use any BRM default or custom event defined in your system.

   The default **pin_notify_telco** file includes the following lines, which indicate that PCM_OP_TCF_PROV_CREATE_SVC_ORDER (opcode number 4016), PCM_OP_TCF_PROV_HANDLE_SVC_ORDER (opcode number 4017), and PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT (opcode number 4019) are called whenever these notification events occur:

   ```
   4016    0       /event/notification/service/pre_create
   4016    0       /event/notification/service/create
   4016    0       /event/notification/service/pre_change
   4016    0       /event/notification/service/post_change
   4016    0       /event/device/associate
   4016    0       /event/device/disassociate
   4016    0       /event/device/replace
   4016    0       /event/notification/profile/pre_modify
   4016    0       /event/notification/profile/modify
   4016    0       /event/device/state
   4017    0       /event/provisioning/service_order/telco
   4017    0       /event/provisioning/service_order/telco/gsm
   4017    0       /event/provisioning/service_order/telco/gsm/telephony
   4017    0       /event/provisioning/service_order/telco/gsm/data
   4017    0       /event/provisioning/service_order/telco/gsm/fax
   4017    0       /event/provisioning/service_order/telco/gsm/sms
   4017    0       /event/provisioning/service_order/telco/gprs
   4019    0       /event/provisioning/service_order/telco
   4019    0       /event/provisioning/service_order/telco/gsm
   4019    0       /event/provisioning/service_order/telco/gsm/telephony
   4019    0       /event/provisioning/service_order/telco/gsm/data
   4019    0       /event/provisioning/service_order/telco/gsm/fax
   4019    0       /event/provisioning/service_order/telco/gsm/sms
   4019    0       /event/provisioning/service_order/telco/gprs
   ```

4. Save and close the file.

5. Load your final event notification list (pin_notify_file) into the BRM database by using the **load_pin_notify** utility:

   ```
   load_pin_notify pin_notify_file
   ```

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

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Specifying the Details to Add to the Service Order

You specify the services and associated devices (if any) to include in a provisioning service order by editing the **pin_telco_provisioning** file. You load the file into the BRM database's **/config/telco/provisioning** object by using the **load_pin_telco_provisioning** utility.

> **Note:** Including associated devices in a service order is not the same as pre-provisioning devices, although you use the **pin_telco_provisioning** file for both operations. For more information, see "About SIM Card Pre-Provisioning".

To specify the service, device, and profile object fields to add to service orders:

1. Open the *BRM_Home*/**sys/data/config/pin_telco_provisioning** file in a text editor.
2. Add the following lines for each *service* that you want to provision:

   ```
   Service provisioning info:
   ServiceType, ProvAction,
   Field1,
   Field2,
   Field3
   ```

   where:

   - *ServiceType* specifies the type of service that is being provisioned.
   - *ProvAction* specifies the provisioning action that is sent in the service order. The external provisioning system uses this information to determine the appropriate provisioning action. Use **A** (activate), **C** (close), **D** (deactivate), **I** (ignore), **P** (provisioning), **R** (reactivate), and **S** (suspend).
   - *FieldX* specifies the service object fields to include in the service order.

   For example, the following lines specify to add the bearer service name, APN name, and QOS profile name to the service order when a GPRS service is being activated. Note that the fields specified are part of the **/service/telco/gprs** schema.

   ```
   Service provisioning info:
   /service/telco/gprs, A,
   PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE,
   PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
   PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME
   ```

3. (Optional) Add the following lines to specify devices associated with the service. Each device you include requires a separate line, separated by commas.

   ```
   Device provisioning info:
   DeviceType, ProvAction, Field1, "String1",
   DeviceType, ProvAction, Field2, "String2"
   ```

where:

- *DeviceType* specifies the type of device associated with the service.
- *ProvAction* specifies the provisioning action for the device objects in the service order. The external provisioning system uses this information to determine the appropriate provisioning action. Use **A** (activate), **C** (close), **D** (deactivate), **I** (ignore), **P** (provisioning), **R** (reactivate), and **S** (suspend).
- *FieldX* specifies the name of a field that contains a device attribute to include in the service order. Field names are replaced by actual values when you run **pin_telco_provisioning**.
- *StringX* specifies a string that is used in the service order to identify the attribute specified by the *field* value. You can query for the string in the service order.

For example, the following line activates a phone number. The actual phone number value will be included in the service order, identified by the MSISDN string.

```
/device/num, A, PIN_FLD_DEVICE_ID, "MSISDN"
```

4. Save and close the file.
5. Run the following command, which loads the file into the database:

   ```
   load_pin_telco_provisioning pin_telco_provisioning
   ```

   For the complete command syntax, see "load_pin_telco_provisioning".
6. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the data was loaded, you can display the **/config** objects by using the Object Browser or use the **robj** command with the **testnap** utility. See **testnap** and "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Specifying the Available States for Each Service Order

You specify the available service order state transitions for each service type by editing the **pin_telco_service_order_state** file. You then load the file into the BRM database's **/config/telco/service_order_state/*** object by using the **load_pin_telco_service_order_state** utility.

To specify the state transitions:

1. Open the *BRM_Home***/sys/data/config/pin_telco_service_order_state** file in a text editor.
2. Add the following lines for each service type that you want to provision:

   ```
   ServiceType
   StateID: StateType: OpcodeNum: Flags
              NextState:OpcodeNum:Flags
              NextState:OpcodeNum:Flags
              NextState:OpcodeNum:Flags
   ```

   where:

   - *ServiceType* specifies the service type that is being provisioned.

- *StateID* specifies the starting service order state: NEW (**1**), READY (**2**), PROCESSING (**3**), COMPLETED (**4**), and FAILED (**5**).
- *StateType* specifies the state type: raw (**0**), init (**1**), normal (**2**), and end (**3**).
- *NextState* specifies the state to which the service order can be transitioned to from the starting state: NEW (**1**), READY (**2**), PROCESSING (**3**), COMPLETED (**4**), and FAILED (**5**).
- *OpcodeNum* specifies the opcode to call when the transition is made. To not call an opcode, use **0**.
- *Flags* specifies the flag to pass to the opcode.

For example, the following lines specify that service orders can transition from a NEW state to a READY state, from a READY state to a PROCESSING state, and from a PROCESSING state to a COMPLETED or FAILED state:

```
/event/service_order/telco/gsm
1: 1: 0: 0
      2: 0:0
2: 2: 0: 0
      3: 0:0
3: 3: 0: 0
      4: 0:0
      5: 0:0
```

3. Save and close the file.
4. Run the following command, which loads the file into the database:

   ```
   load_pin_telco_service_order_state pin_telco_service_order_state
   ```

   For the complete command syntax, see "load_pin_telco_service_order_state".
5. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the data was loaded, you can display the **/config** objects 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*.)

Configuring Service Status Change for Device-to-Service Associations

By default, when you associate a device with a service, BRM activates the service, provisions the associated supplementary features, updates the status of the service and the associated supplementary features, and generates a service order that contains the service status and the status of the associated supplementary features. When you disassociate a device from a service, BRM deactivates the service, unprovisions the associated supplementary features, updates the status of the service and the associated supplementary features, and updates the service order.

You can configure BRM to not update the status of a service when you associate a device with a service or disassociate a device from a service by modifying a field in the **TCF** instance of the **/config/business_params** object.

You modify the **/config/business_params** object by using the **pin_bus_params** utility. For information on this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To configure service status change for device-to-service associations:

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 XML file from the **TCF** instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsTCF bus_params_TCF.xml
   ```

   This command creates the XML file named **bus_params_TCF.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_TCF.xml.out** file in a text editor.

4. Search for the following line:

   ```
   <RestrictDeviceToServiceStatePropagation>disabled</RestrictDeviceToServiceState
   Propagation>
   ```

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

6. Save the file as **bus_params_TCF.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_TCF.xml**

   where *PathToWorkingDirectory* is the directory in which the **bus_params_TCF.xml** file resides.

   Caution: BRM uses the XML in this file to overwrite the existing **TCF** 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 Telco Framework (Services Framework) 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 Connection Manager (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*.

Setting Up Services Framework for Non-Telco Services

To set up Services Framework to process non-telco service types, perform the following tasks:

- Specify the non-telco service types supported by Services Framework. See "Specifying the Non-Telco Services Supported by Services Framework".
- (Optional) Customize the service order details and provisioning mode. See "Customizing the Provisioning Mode Based on Service Order Attributes".
- Set the timeout value for the Confirmed provisioning mode. See "Setting a Timeout Value for Requests Sent in Confirmed Mode".

Specifying the Non-Telco Services Supported by Services Framework

You specify the non-telco service types that are supported by Services Framework by editing the **pin_service_framework_permitted_service_types.xml** configuration file. You then load the XML file into the BRM database's **/config/service_framework/permitted_service_types** object by using the **load_pin_service_framework_permitted_service_types** utility.

The **pin_service_framework_permitted_service_types.xml** file specifies the following for each supported service type:

- The provisioning configuration object, which defines the fields to include in a service order.
- The service order configuration object, which specifies the service order state transitions.
- The provisioning mode: Queued or Confirmed.

> **Note:** You use the XML file primarily to configure your *non-telco* services, but you can use it to specify the provisioning mode for telco services. To do this, list only the telco service type and the provisioning mode.

To specify the supported non-telco service types, perform these tasks:

1. Open the *BRM_Home***/sys/data/config/pin_service_framework_permitted_service_types.xml** file in an XML editor.
2. For each non-telco service type that is supported by Services Framework, add the following entries:
   a. Specify the supported non-telco service type on the **ServiceType** line. For example, replace **ServiceType** with **/service/cable**.

      ```
      <ServiceType>ServiceType</ServiceType>
      ```

   b. Specify the provisioning configuration object to use on the **ConfigTypeProvisioningDetails** line. For example, replace **ProvDetails** with **/config/telco/provisioning/cable**.

      ```
      <ConfigTypeProvisioningDetails>ProvDetails</ConfigTypeProvisioningDetails>
      ```

      For information, see "About Adding Details to the Service Order".

   c. Specify the service order configuration object to use on the **ConfigTypeServiceOrderState** line. For example, replace **SOstate** with **/config/telco/service_order_state/cable**.

      ```
      <ConfigTypeServiceOrderState>SOstate</ConfigTypeServiceOrderState>
      ```

      For information, see "About the Allowable Service Order State Transitions".

d. Specify the provisioning mode on the **ProvisioningMode** line. Replace **Value** with **0** for Queued mode and with **1** for Confirmed mode.

```
<ProvisioningMode>Value</ProvisioningMode>
```

3. Save and close the file.
4. Run the following command, which loads the list of non-telco service types into the database:

```
load_pin_service_framework_permitted_service_types
```

See "load_pin_service_framework_permitted_service_types" for more information.

> **Note:** This utility requires 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*.

5. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the data loaded successfully, you can display the **/config/service_framework/permitted_service_types** 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*.

Customizing the Provisioning Mode Based on Service Order Attributes

By default, Services Framework assigns provisioning modes to events based on the service type. You can customize Services Framework to assign the provisioning mode based on other attributes, such as service order details, by customizing the PCM_OP_TCF_POL_PROV_HANDLE_SVC_ORDER policy opcode. You can also customize the policy opcode to modify service order event details before they are published to the external provisioning agent.

You customize the policy opcode to find events with specific flist fields, assign the appropriate provisioning tag and change service order details, and then return the provisioning mode in the PIN_FLD_MODE output flist field.

For information about customizing policy opcodes, see "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.

Setting a Timeout Value for Requests Sent in Confirmed Mode

When provisioning service orders in Confirmed mode, **dm_prov_telco** waits for a response from the external provisioning agent before activating the service in the BRM database. If the external provisioning agent encounters an error or fails, **dm_prov_telco** might have to wait indefinitely. To prevent this problem, you can configure **dm_prov_telco** to close the connection and roll back the transaction if the wait exceeds a specified amount of time.

To specify a timeout value:

1. Open the **dm_prov_telco** configuration file (*BRM_Home*/**sys/dm_prov_telco/pin.conf**) in a text editor.
2. Set the **prov_timeout** entry to the amount of time, in seconds, that **dm_prov_telco** waits for a response from the external provisioning agent before timing out.

For example, to set the timeout value to 30 seconds, enter the following:

```
-dm_provision prov_timeout 30
```

By default, the timeout value is set to **20**. If you specify **0**, **dm_prov_telco** waits an infinite amount of time for the external provisioning agent to respond.

3. Save and close the file.
4. Stop and restart **dm_prov_telco**. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Enabling In-Flight Changes to Service Orders

By default, Services Framework provisioning generates only one service order for a particular service at a time. For example, when a customer adds a service, such as GPRS telephony, Services Framework provisioning generates a service order to activate the service. Services Framework does not generate any other service orders for that GPRS telephony service until the external provisioning agent sends a response and Services Framework updates the GPRS telephony service's status in the BRM database. This prevents the external provisioning agent from overwriting service changes or processing service orders out of order.

You can enable Services Framework provisioning to generate multiple service orders for a particular service before it receives a response from the external provisioning agent and updates the service's status. This allows you to make in-flight changes to a service's provisioning request. For example, one service order could activate a GPRS telephony service and a second service order could correct the device ID associated with the service.

Note: When Services Framework sends multiple service orders for a particular service to the external provisioning agent, it waits until it has received a response for all requests before updating the service's status in the BRM database.

To enable Services Framework provisioning to generate new service orders for a service that already has a provisioning request in process:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) in a text editor.
2. Add the following entry to the file:

   ```
   - fm_tcf support_multiple_so 1
   ```

   where:

   - **0** prohibits Services Framework provisioning from generating new service orders for a service that already has a provisioning request in process. This is the default.
   - **1** allows Services Framework provisioning to generate service orders for services that already have a provisioning request in process.
3. Save and close the file.
4. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Service and Device Object Updates

Use the PCM_OP_TCF_PROV_UPDATE_PROV_OBJECT opcode to update the status of a service order, based on the **/config/telco/service_order_state** object. Based on the updated service order, this opcode updates the status of the corresponding service, profile, or device object and possibly sub-statuses, such as those for supplementary services.

20

About Performing AAA for Prepaid Services

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) Services Framework AAA Manager and describes how to implement AAA functionality for any service type.

Before you read this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

About Processing AAA Requests for Prepaid Services

Services Framework AAA Manager performs authentication, authorization, and accounting (AAA) for prepaid services. For example, when a prepaid customer uses a service, Services Framework AAA Manager performs the following:

- Verifies the customer's identity by using the device ID.
- Determines whether the customer's prepaid balance contains enough resources to cover the cost of usage.

  If you enabled in-session notifications, Services Framework AAA Manager provides in-session notifications in the AAA responses to your supported network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".
- Reserves a portion of the customer's resources for the session.

  If you enabled in-session notifications, Services Framework AAA Manager provides in-session notifications in the AAA responses to your supported network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".
- Records information about any usage while it is in progress.
- When a session ends, rates the customer's usage and updates the customer's prepaid balance.

  If you enabled in-session notifications, Services Framework AAA Manager provides in-session notifications in the AAA responses to your supported network connectivity application. See "Providing In-Session Notifications for Network Connectivity Applications".

For more information about how BRM performs AAA, see "Understanding Prepaid AAA".

About Services Framework AAA Manager

Services Framework AAA Manager is a generic framework of opcodes, storable classes, and utilities that allows you to quickly implement AAA support for any service type. BRM service managers, such as GSM AAA Manager and GPRS AAA Manager, as well as custom service managers are built on top of the generic framework.

About Services Framework AAA Opcodes

Services Framework AAA opcodes are abstract, generic opcodes designed to process AAA requests for any service type. They perform AAA operations that are common to all service types, such as searching for storable objects and passing information to the Activity FM standard opcodes, and rely upon helper opcodes to perform service-specific functions. For example, helper opcodes build search templates and aggregate service-specific data.

Services Framework AAA standard opcodes call helper opcodes during any of these stages in the opcode's execution:

- SEARCH_SESSION
- PREP_INPUT
- VALIDATE_LIFECYCLE
- ACC_ON_OFF_SEARCH
- TAG_SESSION
- POST_PROCESS

At each processing stage, the opcodes determine which helper opcode to call by reading the **/config/opcodemap/tcf** object. By default, the opcodes call the Services Framework helper opcodes. See "Preparing Service-Specific Data by Using Helper Opcodes".

You configure the Services Framework AAA opcodes to call helper opcodes by using the "load_aaa_config_opcodemap_tcf" utility. See "Configuring Services Framework to Call Helper Opcodes".

About the Services Framework AAA Storable Classes

BRM stores information about prepaid sessions in these storable classes.

- **/active_session/telco**: Stores information about prepaid sessions that *are in progress*. This object can be subclassed for specific service types.
- **/event/session/telco**: Stores information about a prepaid session that has been *rated and closed*. This object can be subclassed for specific service types.
- **/reservation**: When your system does not include IMDB Cache, stores information about a single reservation for one balance group.

  For policy-driven sessions, this object holds the consumed reservation amount for the resource.
- **/reservation/active**: When your system includes IMDB Cache, stores information about a single reservation for one balance group.
- **/reservation_list**: Tracks the total resources a balance group has reserved in **/reservation/active** objects.

- **/config/reserve**: Stores the default authorization and reauthorization values for specific service types. This object can be subclassed for specific service types.

  See "Specifying Default Authorization and Reauthorization Values" for information on how to load this object.

- **/config/aaa**: Stores configuration information on how to manage session objects. See "Configuring How Services Framework AAA Manages Session Objects" for information on how to load this object.

  Stores the maximum value permitted for the scaled delay for a resource to be used when providing the tariff change indication in the in-session notifications to network connectivity applications. See "About Configuring the Maximum Scaled Delay Time for a Resource".

- **/config/opcodemap/tcf**: Stores the mappings between Services Framework AAA opcodes and helper opcodes. See "Configuring Services Framework to Call Helper Opcodes" for information on how to load this object.

About the Services Framework AAA Manager Utilities

You use Services Framework AAA Manager utilities to specify default preferences for processing events, including the following:

- How to store and rate accounting subsessions.
- Whether to keep or delete **/active_session** and **/reservation/active** objects when a prepaid session ends.
- Whether to check for duplicate **/active_session** or **/event/session** objects.
- The expiration time interval for **/active_session** objects stored in memory.
- Default authorization and reauthorization values for services. BRM authorizes prepaid customers to use a service for a specified duration, volume, or amount. For example, BRM can authorize customers to initially make a 10 minute telephone call or download 100 bytes of data.

See "Specifying Default AAA Preferences".

Utilities to Load Data Specific to In-Session Notifications

If you enabled in-session notifications, you use Services Framework AAA Manager utilities to specify default preferences for the following:

- The maximum scaled delay time by resource ID. See "About Configuring the Maximum Scaled Delay Time for a Resource".
- The subscriber preferences for in-session notifications. See "Customizing Subscriber Preferences".

Services Framework AAA Manager Process Overview

When a service manager calls a Services Framework AAA opcode, the opcode performs the following:

1. Retrieves the customer's account information.
2. At the SEARCH_SESSION processing stage, calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode returns a search template for finding **/event/session** or **/active_session** objects.
3. Uses the search template returned by the helper opcode to find duplicate sessions.

4. At the PREP_INPUT stage, calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode aggregates service-specific data and prepares a service-specific input flist.
5. At the VALIDATE_LIFECYCLE stage, calls the helper opcode specified in the **/config/opcodemap/tcf** object to validate the request if the service uses a custom life cycle. If validation succeeds, processing continues. If validation fails, the request is denied.
6. Passes the input flist returned by the helper opcode to an Activity FM standard opcode. The Activity opcode rates or records the event and then returns that the request succeeded or failed.
7. At the POST_PROCESS stage, calls the helper opcode specified in the **/config/opcodemap/tcf** object. The helper opcode aggregates service-specific data and returns a service-specific input flist.
8. Returns to the caller that the request either succeeded or failed.

About Tracking Data in Master Sessions and Subsessions

A customer may use multiple services during a call connection. For example, a customer may send text messages and access email during one connection. The external network connects the service by opening a PDP context and then begins collecting usage information. The external network sends the usage data to BRM, which stores all of the information in one session object.

You can set up your BRM system to track and rate the usage for each service separately by using master sessions and subsessions. The master session tracks the usage for the first service accessed during the connection. Each subsession tracks the usage for each subsequent service that is accessed during the connection. For example, if a customer sends a text message and then accesses email during one connection, BRM stores the text message usage data in a master session and the email usage data in a subsession. This enables BRM to rate each service separately, based on the service's specific rating criteria.

To track data in master sessions and subsessions, you must:

- Configure your external network to call BRM each time a master session or subsession starts. It must also call BRM each time a network trigger, such as a quality of service (QoS) change, occurs.
- Configure your external network to flag each session as a master session or a subsession.
- (Optional) Set up your system to store master session and subsession data in separate storable class extensions.
- Configure your external network to notify BRM when the last subsession has ended.
- Configure how BRM manages and rates master sessions and subsessions.

Specifying How to Rate Subsessions

If a session includes both a master session object and one or more subsession objects, BRM performs one of the following at the end of the session:

- Aggregates the data from the master and subsession objects into one session object.

- Stores the data for each master and subsession object separately.

You specify how BRM manages subsession data by setting the subsession rating mode.

Table 20–1 describes how each rating mode stores and rates subsession data:

Table 20–1 Rating Modes and Subsession Data

Rating Mode	How Subsession Data is Stored	When a Subsession is Rated and Closed	Description
Rate mode	In separate subsession objects.	When *each* subsession ends.	Master and subsessions are treated as separate AAA sessions and are rated separately. The master and subsession objects are created on receipt of the authorization request. The master and subsession objects are closed and rated, and the event is recorded on receipt of each subsession's stop accounting request.
Deferred rate mode	In separate subsession objects.	When *the last* subsession ends.	Master and subsessions are treated as separate AAA sessions and are rated separately. The master and subsession objects are created on receipt of the authorization request. The master and subsession objects are rated, and the event is recorded when the final stop accounting request is received.
Aggregate mode	In one master session object.	When *the last* subsession ends.	Usage information for subsessions is aggregated and recorded in a single usage event. During authorization of subsessions, new **/active_session** and **/reservation** objects are created. During stop accounting of subsessions, all of the **/active_session** objects are aggregated and a single event is created. **Note:** By default, only volume and amount are aggregated.

For example, a customer connects to the network to download email and then in parallel wants to view a movie extract during the same connection. Because the customer uses the same connection for both services, BRM treats the usage as one session. However, because the email download may finish before the streaming session ends, BRM can optionally record the usage in two separate objects: a master session object that tracks the volume of email downloaded, and a subsession object that tracks both the volume and duration of the streaming session. If the rating mode is set to deferred rate mode, BRM waits until both the email download and streaming session have ended and then rates both usages at the same time.

You configure the rating mode by using the **load_pin_telco_aaa_params** utility. See "Configuring How Services Framework AAA Manages Session Objects".

Flagging Session Data As a Master Session or a Subsession

When a customer accesses multiple services during a connection, BRM, by default, treats all the usage data as one AAA session.

To flag session data as a master session or a subsession, configure your external network to pass the following two input flist fields in the call to the Services Framework AAA opcodes:

- PIN_FLD_SESSION_TYPE set to **0** for a normal session, **1** for a master session, or **2** for a subsession. The default is **0**.
- PIN_FLD_NETWORK_SESSION_CORRELATION_ID set to the unique session identifier. The master session and all of its subsessions must reference the same identifier.

For example, a master session would include these input flists fields:

```
0 PIN_FLD_SESSION_TYPE                     ENUM [0] 1
0 PIN_FLD_NETWORK_SESSION_CORRELATION_ID   STR [0] "MA_1"
```

Any subsessions that started during the same connection would include these input flist fields:

```
0 PIN_FLD_SESSION_TYPE                     ENUM [0] 2
0 PIN_FLD_NETWORK_SESSION_CORRELATION_ID   STR [0] "MA_1"
```

Specifying to Store Master and Subsession Data in Storable Class Extensions

By default, BRM stores information about a master session and any of its subsessions in one storable class type: **/active_session/telco/***. For example, if a session includes a master session and two subsessions and the subsession mode is rate mode or deferred rate mode, BRM creates three **/active_session/telco** objects.

You can direct BRM to store master session data in one storable class extension and subsession data in another storable class extension through calls to the Services Framework AAA opcodes. For example, BRM could store GPRS master session data in an **/active_session/telco/gprs/master** object. Likewise, BRM could store GPRS subsession data in an **/active_session/telco/gprs/subsession** object.

To store session data in a storable class extension, configure your external network to pass the PIN_FLD_OBJ_TYPE input flist field in the call to the Services Framework AAA opcodes. Set the PIN_FLD_OBJ_TYPE input flist field to the extension that should be added after **/active_session/telco**. For example, to store master session data in an **/active_session/telco/gprs/master** object:

```
0 PIN_FLD_OBJ_TYPE    STR [0] "gprs/master"
```

Specifying Whether a Network Connection Is Closed or Still Open

When the rating susbsession mode is set to deferred rate mode or aggregate mode, BRM waits until the master session and all subsessions have ended before rating the usage data. In this case, the external network must notify BRM whether the network connection is closed or still open when a subsession ends.

Your external network indicates whether a network connection has ended or is still open by passing the PIN_FLD_SESSION_STOP_INDICATOR input flist field in the call to the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode. Set the PIN_FLD_SESSION_STOP_INDICATOR field to one of the following:

- **0** to specify that the network session is still in progress.
- **1** to specify that the network session has finished.

Ensuring That All Subsessions Have Stopped before Closing the Master Session

When AAA opcodes receive the final stop accounting request, the opcodes, by default, confirm that the master session and all subsessions have ended and have been processed before closing all of the **/active_session** objects and creating events. This prevents revenue leakage when subsessions are processed out of order.

To disable the extra search, perform these steps:

1. Open the Services Framework AAA configuration file (*BRM_Home*/**apps/tcf_aaa/pin.conf**) in a text editor.
2. Set the **wait_for_all_interim_stop_request** entry to **NO**:
   - When set to **YES**, the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode confirms that all stop accounting requests have been processed after it receives

the final stop accounting request. This ensures that all usage information is captured if subsessions are processed out of order. This is the default setting.

- When set to **NO**, the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode does not confirm that all stop accounting requests have been processed after it receives the final stop accounting request.

```
- fm_tcf_aaa    wait_for_all_interim_stop_request    NO
```

3. Save and close the file.
4. Stop and restart the Services Framework AAA Manager.

Specifying Default AAA Preferences

You can specify how BRM processes AAA requests for services by using the Services Framework AAA Manager utilities and configuration files. For example, you can specify the following:

- The default authorization and reauthorization values for services. "Specifying Default Authorization and Reauthorization Values".
- How to handle session objects. See "Configuring How Services Framework AAA Manages Session Objects".
- The helper opcodes to call. See "Configuring Services Framework to Call Helper Opcodes".
- How to calculate the total resources reserved by an account. See "Configuring How BRM Calculates Reservation Balances".

Specifying Default Authorization and Reauthorization Values

Prepaid customers are authorized to use a service for a specified amount, duration, volume, or activity. For example, you can authorize customers to initially download 100 bytes of data or make a 30-minute telephone call.

By default, BRM authorizes customers for the volume or duration passed in the input flist to the AAA opcodes. If a value is not passed in, the opcode uses the default value specified in the service-specific **/config/reserve** object. For example, it uses the **/config/reserve/gsm/data** object for GSM data services. If there is not an object for the specified service type, the opcode uses the default configuration object (**/config/reserve**).

You define default values by using the "load_config_reservation_aaa_prefs" utility. This utility loads the values from one of the following reservation preferences files into a service-specific **/config/reserve** object in the BRM database:

- **pin_config_reservation_aaa_prefs**
- **pin_config_reservation_aaa_prefs_gprs**
- **pin_config_reservation_aaa_prefs_gsm**
- **pin_config_reservation_aaa_prefs_gsm_data**
- **pin_config_reservation_aaa_prefs_gsm_telephony**
- **pin_config_reservation_prefs**

The reservation preferences file specifies the default values in flist format.

Single-RUM example:

```
0 PIN_FLD_RESERVATION_INFO          ARRAY [0]
1 PIN_FLD_QUANTITY                DECIMAL [0] 100
1 PIN_FLD_MIN_QUANTITY            DECIMAL [0] 0
1 PIN_FLD_INCR_QUANTITY           DECIMAL [0] 100
1 PIN_FLD_RUM_NAME                    STR [0] "Amount"
1 PIN_FLD_IS_PRIMARY_RUM             ENUM [0] 1
1 PIN_FLD_REQ_MODE                   ENUM [0] 1
1 PIN_FLD_UNIT                       ENUM [0] 0
1 PIN_FLD_RATIO                       INT [0] 2
```

Multi-RUM example:

```
0 PIN_FLD_RESERVATION_INFO       ARRAY [0]
1 PIN_FLD_QUANTITY             DECIMAL [0] 100
1 PIN_FLD_MIN_QUANTITY         DECIMAL [0] 0
1 PIN_FLD_INCR_QUANTITY        DECIMAL [0] 100
1 PIN_FLD_RUM_NAME                 STR [0] "Amount"
1 PIN_FLD_REQ_MODE                ENUM [0] 1
1 PIN_FLD_UNIT                    ENUM [0] 0
1 PIN_FLD_IS_PRIMARY_RUM          ENUM [0] 0
1 PIN_FLD_RATIO                    INT [0] 1
0 PIN_FLD_RESERVATION_INFO       ARRAY [1]
1 PIN_FLD_QUANTITY             DECIMAL [0] 50
1 PIN_FLD_MIN_QUANTITY         DECIMAL [0] 0
1 PIN_FLD_INCR_QUANTITY        DECIMAL [0] 50
1 PIN_FLD_REQ_MODE                ENUM [0] 2
1 PIN_FLD_UNIT                    ENUM [0] 0
1 PIN_FLD_IS_PRIMARY_RUM          ENUM [0] 1
1 PIN_FLD_RATIO                    INT [0] 2
0 PIN_FLD_RESERVATION_INFO       ARRAY [2]
1 PIN_FLD_QUANTITY             DECIMAL [0] 5
1 PIN_FLD_INCR_QUANTITY        DECIMAL [0] 50
1 PIN_FLD_RUM_NAME                 STR [0] "Volume"
1 PIN_FLD_REQ_MODE                ENUM [0] 4
1 PIN_FLD_UNIT                    ENUM [0] 0
1 PIN_FLD_IS_PRIMARY_RUM          ENUM [0] 0
1 PIN_FLD_RATIO                    INT [0] 4
```

- PIN_FLD_RESERVATION_INFO is an array that stores the default authorization and reauthorization values for one ratable usage metric (RUM). You create an array for each RUM that you support.
- PIN_FLD_QUANTITY specifies the default authorization quantity.
- PIN_FLD_MIN_QUANTITY specifies the minimum quantity to be rated for an AAA session. For example, you can specify a minimum of 1 minute or 1 MB is rated for an AAA session.
- PIN_FLD_INCR_QUANTITY specifies the default extension quantity when reauthorizing a prepaid session. For example, you can specify to reauthorize a customer for an additional 20 minutes or 100 MB.
- PIN_FLD_RUM_NAME specifies the ratable usage metric (RUM) to use when calculating the authorization and reauthorization value.
- PIN_FLD_IS_PRIMARY_RUM specifies whether this RUM is a primary RUM. **0** indicates that this is not a primary RUM; **1** indicates that this is a primary RUM. This field is optional. It is ignored if present for a single-RUM request.
- PIN_FLD_REQ_MODE specifies to rate the amount (**1**), duration (**2**), volume (**4**), duration and volume (**6**), or occurrence (**8**).

- PIN_FLD_UNIT specifies the currency resource ID for amounts; for example, 840 for US dollars or 978 for Euros. This field is required only for amounts.
- PIN_FLD_RATIO specifies the ratio for scaling in multi-RUM scenarios. This field is optional. It is ignored if present for a single-RUM request.

To specify default authorization and reauthorization values:

1. Open the reservation preferences file in a text editor. Sample files are located in the *BRM_Home*/**sys/data/config** directory.
2. Specify your default authorization and reauthorization values.
3. Save the file.
4. Load the file into the BRM database by using the "load_config_reservation_aaa_prefs" utility.

   ```
   load_config_reservation_aaa_prefs reservation_prefs
   ```

   Note: To connect to the BRM database, this utility requires a configuration file in the directory from which you run it. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

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

To verify that the default authorization and reauthorization values were loaded, you can display the **/config/reserve** 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*.

Configuring How Services Framework AAA Manages Session Objects

You configure how Services Framework AAA manages session objects by editing the appropriate AAA parameters XML file in the *BRM_Home*/**sys/data/config** directory:

- **pin_telco_aaa_params.xml**
- **pin_telco_gprs_aaa_params.xml**
- **pin_telco_gsm_aaa_params.xml**
- **pin_telco_gsm_data_aaa_params.xml**
- **pin_telco_gsm_fax_aaa_params.xml**
- **pin_telco_gsm_sms_aaa_params.xml**
- **pin_telco_gsm_telephony_aaa_params.xml**

You then load the file into the BRM database's **/config/aaa** object by running the **load_pin_telco_aaa_params** utility.

About Maximum Scaled Delay Times for Resources and In-Session Notifications

If you enabled in-session notifications, specify the maximum scaled delay time for each supported resource in the AAA parameters XML files listed above.

When BRM processes a AAA service request for a resource from the network connectivity application, if you have not configured the maximum scaled delay time for that resource in the appropriate AAA parameters XML file, BRM does not calculate

or provide the tariff change indication for that resource in its in-session notifications to the network connectivity application.

See *BRM_Home*/**sys/data/config/config_beid.xml** file for the set of configured resources.

Configuring Services Framework AAA Parameters XML Files

To configure how Services Framework AAA manages session objects:

1. Go to the *BRM_Home*/**sys/data/config** directory and open the appropriate AAA parameters XML file in a text editor.
2. Specify whether BRM checks for duplicate **/active_session** or **/event/session** objects by editing the **DuplicateCheckType** entry:

   ```
   <DuplicateCheckType>1</DuplicateCheckType>
   ```

   - **1** specifies to check for duplicate **/active_session** objects.
   - **2** specifies to check for duplicate **/event/session** objects.
3. Specify how long in seconds to store **/active_session** objects in memory before deleting them by editing the **ExpirationInterval** entry:

   Note: This entry is required for IMDB Cache. This value must match the **ExpirationTimeInSeconds** registry entry value. Set **ExpirationInterval** to **0** if the transient object pool is disabled. Se.

   ```
   <ExpirationInterval>0</ExpirationInterval>
   ```
4. Specify how to store and rate subsessions by editing the **SubsessionMode** entry:

   ```
   <SubsessionMode>1</SubsessionMode>
   ```

   - **1** specifies *aggregate mode*. Subsessions are stored and rated in one event object.
   - **2** specifies *rate immediately*. Subsessions are stored and rated individually. BRM rates each subsession as soon as it ends.
   - **3** specifies *deferred rate mode*. Subsessions are stored and rated individually. BRM rates all of the subsessions at once, after the last subsession ends.

   For more information, see "Specifying How to Rate Subsessions".
5. Specify whether to keep or delete **/active_session** objects and **/reservation/active** objects when a prepaid session ends by editing the **DeletedFlag** entry:

   ```
   <DeletedFlag>3</DeletedFlag>
   ```

   - **0** specifies to keep both **/active_session** and **/reservation/active** objects.
   - **1** specifies to keep **/active_session** objects but delete **/reservation/active** objects.
   - **2** specifies to delete **/active_session** objects but keep **/reservation/active** objects.
   - **3** specifies to delete both **/active_session** and **/reservation/active** objects.
6. Specify the maximum scaled delay times for the supported resources.

If the file contains a **ScaledDelayInfo** entry for a resource, the current maximum scaled delay time for the resource is the **MaxScaledDelayTime** value specified for that **ScaledDelayInfo** entry. Change the entry for **MaxScaledDelayTime**.

If the file does not contain a maximum scaled delay time for a resource, add the **ScaledDelayInfo** entry for that resource and enter its **MaxScaledDelayTime** value. For example,

```
<ScaledDelayInfo ResourceId="978">
<MaxScaledDelayTime>12000</MaxScaledDelayTime>
</ScaledDelayInfo>
```

In this example,

- The resource for which the maximum scaled delay time is being added to the file is **978** (Euro).
- The maximum scaled delay time for this resource is set as *12000* seconds.

7. Save and close the file.
8. Load the AAA parameters XML file into the BRM database by using the **load_pin_telco_aaa_params** utility:

   ```
   load_pin_telco_aaa_params [-f pin_telco_aaa_params_file]
   ```

   where **pin_telco_aaa_params_file** specifies the name and location of the file that defines how to manage session objects. By default, the utility uses the *BRM_Home*/**sys/data/config/pin_telco_aaa_params.xml** file.

   For more information, see "load_pin_telco_aaa_params".
9. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the configuration parameters were loaded, you can display the **/config/aaa** 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*.

Configuring Services Framework to Call Helper Opcodes

You configure Services Framework AAA Manager to call helper opcodes at specific processing stages by using the "load_aaa_config_opcodemap_tcf" utility. This utility loads your mappings from the **pin_config_opcodemap_tcf** configuration file into the **/config/opcodemap/tcf** object in the BRM database.

The **pin_config_opcodemap_tcf** file maps AAA opcodes to helper opcodes in the following format:

```
Framework_Opcode: Opcode_Name
Processing_Stage: Stage
Opcode_Map: Service_Type, Helper_Opcode
```

where:

- *Opcode_Name* specifies the name of the opcode.
- *Stage* specifies the processing stage at which to call the helper opcode. The processing stages are SEARCH_SESSION, PREP_INPUT, VALIDATE_LIFECYCLE, ACC_ON_OFF_SEARCH, TAG_SESSION, and POST_PROCESS.
- *Service_Type* specifies the service type that triggers a call to the helper opcode.

- *Helper_Opcode* specifies the name of the helper opcode to call. For each combination of framework opcode, processing stage, and service type, Services Framework can call only one helper opcode.

  Note: You can configure the dropped calls feature to call more than one helper opcode during the TAG_SESSION stage. See "Setting Up Your System to Identify Dropped Calls and Continuation Calls" for more information.

For example, the following entry specifies that when the PCM_OP_TCF_AAA_AUTHORIZE opcode processes **/service/telco/gsm/sms** services and reaches the PREP_INPUT processing stage, it calls the PCM_OP_TCF_AAA_AUTHORIZE_PREP_INPUT helper opcode.

```
Framework_Opcode: PCM_OP_TCF_AAA_AUTHORIZE
Processing_Stage: PREP_INPUT
Opcode_Map:/service/telco/gsm/sms, PCM_OP_TCF_AAA_AUTHORIZE_PREP_INPUT
```

To configure opcodes to call helper opcodes:

1. Open the *BRM_Home*/**sys/data/config/pin_config_opcodemap_tcf** file in a text editor.
2. Edit the file.
3. Save and close the file.
4. Load the file into the BRM database by using the **load_aaa_config_opcodemap_tcf** utility.

   Note: To replace the entire contents of the **/config/opcodemap/tcf** 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
   ```

   For more information, see "load_aaa_config_opcodemap_tcf".
5. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that your opcode mappings were loaded, you can display the **/config/opcodemap/tcf** 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*.

Configuring How BRM Calculates Reservation Balances

BRM tracks the total resources reserved by a balance group in two objects:

- **/reservation_list**: This object tracks the total resources a balance group has reserved in **/reservation/active** objects. This object is required for IMDB Cache-enabled prepaid systems.

  For policy-driven charging sessions, this object also holds the consumed reservation amount for those resources.

- **/balance_group**: This object's RESERVED_AMOUNT field tracks the total resources a balance group has reserved in **/reservation** objects.

  For policy-driven charging sessions, this object also holds the consumed reservation amount for those resources.

You specify how BRM calculates a customer's reservation balance by setting the **balance_coordinator** entry in the CM's **pin.conf** file.

- When the entry is set to **0**, BRM retrieves reservation balances from both types of objects. Use this setting for systems that have the CM connected to IMDB Cache DM.
- When the entry is set to **1**, BRM retrieves reservation balances from **/balance_group** objects only. Use this setting for systems that have the CM connected to Oracle DM. This is the default setting.

Calculating Reservation Balances for IMDB Cache-Enabled Systems

For IMDB Cache-enabled prepaid systems, you must configure BRM to calculate reservation balances by adding the balances from the **/balance_group** and **/reservation_list** objects:

1. Open the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **balance_coordinator** entry to **0**:

   ```
   - fm_bal balance_coordinator 0
   ```

3. Save and close the file.

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

Calculating Reservation Balances for Non-IMDB Cache Systems

For prepaid systems that do not use IMDB Cache, you must configure BRM to calculate reservation balances by retrieving the RESERVED_AMOUNT field in the **/balance_group** object:

1. Open the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**) in a text editor.
2. Set the **balance_coordinator** entry to **1**.

   ```
   - fm_bal balance_coordinator 1
   ```

3. Save and close the file.
4. Make sure you pass the PIN_FLD_RESERVATION_OBJ input flist field for the following opcodes:
   - PCM_OP_TCF_AAA_AUTHORIZE
   - PCM_OP_TCF_AAA_REAUTHORIZE
   - PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE

   Note: The PIN_FLD_RESERVATION_OBJ field is marked as optional in the opcode input flist. However, this field is mandatory for prepaid systems that do not use IMDB Cache for AAA.

Improving Search Performance for Prepaid Services

By default, BRM locates a customer's service object by using the customer's login name and then, if the login name is not found, by using the customer's alias name. That is, BRM searches through **/service** objects by first using the object's PIN_FLD_LOGIN field and then using the object's PIN_FLD_ALIAS_LIST array.

However, prepaid service types use alias names rather than login names to identify customers. This increases the amount of time required to find these **/service** objects.

You can improve search performance for prepaid service types by configuring BRM to search alias names first and login names second for specified service types. To do this, add the list of service types that identify users through alias names to the **pin_excluded_logins.xml** file and then load it into the BRM database's **/config/login_exclusion** object.

To improve search performance for services, perform the following tasks:

1. Open the *BRM_Home***/sys/data/config/pin_excluded_logins.xml** file in a text editor.
2. Add an entry for each service type that identifies users through alias names rather than login names.

   Note: The default XML file contains the following entries, but you can add custom services to the file.

```
<LoginExclusionManagementComponent>
   <ServiceList>
      <Service>/service/telco/gsm</Service>
      <Service>/service/telco/gsm/telephony</Service>
      <Service>/service/telco/gsm/fax</Service>
      <Service>/service/telco/gsm/data</Service>
      <Service>/service/telco/gsm/sms</Service>
   </ServiceList>
</LoginExclusionManagementComponent>
```

3. Save and close the file.
4. Load the XML file into the BRM database by using the **load_pin_excluded_logins** utility.

   Note: This utility requires a configuration file in the directory from which it is run.

```
cd BRM_Home/sys/data/config
load_pin_excluded_logins pin_excluded_logins.xml
```

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

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

To verify that the **pin_excluded_logins.xml** file loaded properly, display the **/config/login_exclusion** object by using the Object Browser or by using the **robj** command with the **testnap** utility. For information, see "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Configuring Services Framework AAA Manager for Custom RUMs

By default, Services Framework AAA Manager supports the following RUM types: duration, volume, amount, and occurrence. To enable Services Framework AAA Manager to perform AAA by using a custom RUM, you must configure it to do so.

- To perform AAA for a custom RUM, see "Supporting Custom RUMs during the AAA Process".
- To perform AAA for a custom RUM in a multiple RUM scenario, see "Supporting Custom RUMs in a Multiple RUM Scenario".

Supporting Custom RUMs during the AAA Process

To configure Services Framework AAA Manager to rate events by using custom RUMs:

- Define your custom RUMs in the *BRM_Home*/**sys/data/pricing/example/pin_rum** file and then use the **load_pin_rum** utility to load it into the database's **/config/rum** object. See "Setting Up Ratable Usage Metrics (RUMs)" and **load_pin_rum** in *BRM Setting Up Pricing and Rating*.
- Specify the default authorization and reauthorization values for the custom RUM in the reservation preferences file and then use the **load_config_reservation_aaa_prefs** utility to load it into the database's **/config/reserve** object. See "Specifying Default Authorization and Reauthorization Values".

  Note: Make sure you map the new PIN_FLD_REQ_MODE field for the custom RUM.

- Configure your external network to collect the request mode and pass it in the PIN_FLD_REQ_MODE input flist of the appropriate Services Framework AAA opcode. If a mode is not passed in the input flist, Services Framework uses the mode specified in the default **/config/reserve** object.
- For authorization requests, configure your external network to collect the initial authorization quantity and pass it in the PIN_FLD_QUANTITY input flist of the PCM_OP_TCF_AAA_AUTHORIZE opcode. If a quantity is not specified in the input flist, the opcode uses the quantity specified in the default **/config/reserve** object.
- For reauthorization requests as well as update and reauthorization requests:
  - Configure your external network to collect the request mode, extension quantity, and original authorization quantity and pass it in the input flist of the PCM_OP_TCF_AAA_REAUTHORIZE opcode and the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode.
  - Customize the PCM_OP_TCF_AAA_REAUTHORIZE_PREP_INPUT helper opcode to aggregate the custom RUM fields. See "Preparing Service-Specific Flists for Reauthorization".

Supporting Custom RUMs in a Multiple RUM Scenario

Services Framework AAA Manager can perform AAA for GSM and custom service types by using multiple RUMs that consist of the following RUM types only: amount, duration, volume, and occurrence. To perform AAA for GSM or custom service types

by using multiple RUMs that include a custom RUM type, you must perform one of the following:

- Customize the service-specific POST_PROCESS helper opcode to handle the logic for the custom RUM. For example, for GSM services, you would modify the PCM_OP_GSM_AAA_POL_POST_PROCESS helper opcode to aggregate GSM data according to the custom RUM type and the value passed in the PIN_FLD_AGGREGATE_MODE flist field. See "Aggregating Return GSM Data".
- Configure Services Framework AAA Manager to skip all service-specific aggregation during the post-processing stage when processing reauthorization requests for any services that use a custom RUM in a multiple RUM scenario.

To configure Services Framework AAA Manager to skip all service-specific aggregation during the post processing stage when processing reauthorization requests:

1. Open the *BRM_Home*/**sys/data/config/pin_config_opcodemap_tcf** file in a text editor.
2. At the POST_PROCESS stage of the PCM_OP_TCF_AAA_REAUTHORIZE and PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcodes, remove all lines that map a service type that uses your custom RUM to a service-specific helper opcode.

   For example, if your GSM services support a custom RUM in a multiple RUM scenario, you would comment out or delete the entries shown in bold below:

   ```
   #Framework_Opcode: PCM_OP_TCF_AAA_REAUTHORIZE
   #Processing_Stage: POST_PROCESS
   #Opcode_Map:/service/telco/gsm/telephony, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Opcode_Map:/service/telco/gsm/sms, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Opcode_Map:/service/telco/gsm/data, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Opcode_Map:/service/telco/gsm/fax, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Framework_Opcode: PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE
   #Processing_Stage: POST_PROCESS
   #Opcode_Map:/service/telco/gsm/telephony, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Opcode_Map:/service/telco/gsm/sms, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Opcode_Map:/service/telco/gsm/data, PCM_OP_GSM_AAA_POL_POST_PROCESS
   #Opcode_Map:/service/telco/gsm/fax, PCM_OP_GSM_AAA_POL_POST_PROCESS
   ```

3. Save and close the file.
4. Load the file into the BRM database by using the **load_aaa_config_opcodemap_tcf** utility.

   Note: To replace the entire contents of the **/config/opcodemap/tcf** 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
   ```

   For more information, see "load_aaa_config_opcodemap_tcf".
5. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Implementing AAA Functionality for Custom Service Types

To set up your system to perform prepaid AAA for custom service types, perform the following tasks:

1. Set up your external network to collect the information needed and pass the information to the Services Framework AAA standard opcodes. See "Sending AAA Requests to the Services Framework AAA Opcodes".
2. Add a library file for your custom service manager to the *BRM_Home*/**lib** directory and add a pointer to the library file in the CM **pin.conf** file. See "Syntax for Facilities Module (FM) Entries" in *BRM System Administrator's Guide*.
3. Create custom helper opcodes to aggregate service-specific data. You can use the Services Framework AAA helper opcodes as a starting place. See "Preparing Service-Specific Data by Using Helper Opcodes".
4. Configure the Services Framework AAA standard opcodes to call your custom helper opcodes. See "Configuring Services Framework to Call Helper Opcodes".

Sending AAA Requests to the Services Framework AAA Opcodes

To perform AAA for custom services, your system must be designed to collect the information needed from the customer and pass the appropriate fields in the input flist to the Services Framework AAA opcodes.

Your external network can pass information to the opcodes through the following:

- Oracle Communications Service Broker (OCSB).
- A service manager, such as GSM AAA Manager, GPRS Manager, or a custom service manager. See "About Performing AAA for Prepaid GSM Services".

You can use the Services Framework AAA API to perform the following actions:

- Authenticate customers. See "Authenticating Users for Custom Services".
- Authorize customers to use services. See "Authorizing Prepaid Services".
- Reauthorize customers to continue using a service. See "Reauthorizing Prepaid Sessions".
- Update and reauthorize prepaid sessions. See "Updating and Reauthorizing Prepaid Sessions".
- Cancel authorization. See "Canceling Authorization for Prepaid Services".
- Rate and record activity events. See "Rating and Recording Activity Events".
- Manage prepaid sessions. See "Managing Prepaid Sessions".

  If you have enabled in-session notifications, add custom details to the response sent to the network connectivity application. See ""Customizing AAA Processes by Extending Policies".

Authenticating Users for Custom Services

To authenticate users for custom services, call the PCM_OP_TCF_AAA_ AUTHENTICATE opcode with the following information in the input flist:

- Service type
- Device ID
- Action

- (Optional) Password. If a password is passed in, BRM authenticates in PAP mode. If one is not passed in, BRM authenticates in CHAP mode.

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about the authentication process, see "How BRM Authenticates Prepaid Customers".

Authorizing Prepaid Services

The authorization process determines whether a customer is allowed to access a specific service. This includes determining whether the customer has a subscription for the specified service and sufficient resources for the requested service.

Customers that are approved to access a service are authorized for a specified volume or duration and, optionally, a validity period.

For more information, see "About Authorizing Prepaid Usage".

To authorize prepaid customers to use a service, call the PCM_OP_TCF_AAA_AUTHORIZE opcode with the following information in the input flist:

- Service type
- Device ID
- Direction of the call
- Name of the calling application
- Details about the event, such as the IMEI value, the dialed number, and the quality of service (QoS)
- (Optional) The authorization request mode: volume, duration, or amount
- (Optional) The authorization request's volume or duration

> **Note:** If you are not using IMDB Cache for AAA, you must also pass the PIN_FLD_RESERVATION_OBJ field in the opcode's input flist. For more information, see "Calculating Reservation Balances for Non-IMDB Cache Systems".

The opcode determines whether the customer has sufficient resources to use the service for the volume or duration passed in the input flist. If a volume or duration is not passed in, the opcode uses the default authorization value specified in the **/config/reserve** object. (See "Specifying Default Authorization and Reauthorization Values".)

The opcode returns the following to the calling application:

- The result of the authorization (pass or fail), which is returned in the PIN_FLD_RESULT output flist field.
- The authorized volume or duration, which is returned in the PIN_FLD_QUANTITY output flist field.
- (Volume-based authorizations only) The authorization validity period, which is returned in the PIN_FLD_VALID_TO output flist field.

> **Note:** This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about the authorization process, see "How BRM Authorizes Users to Access Prepaid Services".

- If you have enabled in-session notifications and the call request is not denied, the opcode returns additional in-session notifications for credit threshold breaches, service expirations, streaming usage threshold summaries, tariff change indications and subscriber preferences for such notifications.

About Limiting Event Authorization Time

If you are rating with real-time non-duration RUMs, you can use the optional PIN_FLD_VALID_TO output flist field to limit product authorizations to a specific time period (in conjunction with PIN_FLD_RATED_TIMEZONE). This allows you to limit authorizations to the rate tier they are appropriate for, and force a reauthorization of the event when usage crosses into a different rate tier.

By default, if an event is rated with two different products or rate tiers that have two different time limits, the *shortest* time limit is added to PIN_FLD_VALID_TO. There is an exception if the event spans two time periods (for example, starts in off-peak hours and ends in peak hours), and is rated by the event end time. In that case, PIN_FLD_VALID_TO is populated with the end time of the second period.

Reauthorizing Prepaid Sessions

To reauthorize customers to continue their prepaid session, call the PCM_OP_TCF_AAA_REAUTHORIZE opcode with the following information in the input flist:

- Service type
- Device ID
- Direction of the call
- Name of the calling application
- Requested reauthorization amount or quantity
- Details about the call, such as the IMEI value, the dialed number, and the QoS

> **Note:** If you are not using IMDB Cache for AAA, you must also pass the PIN_FLD_RESERVATION_OBJ field in the opcode's input flist. For more information, see "Calculating Reservation Balances for Non-IMDB Cache Systems".

You can specify whether the reauthorization amount or quantity is cumulative or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **4** specifies that the reauthorization amount or quantity passed in the input flist is *cumulative* (that is, it represents the original authorization amount plus a requested extension amount). BRM reauthorizes by using the amount or quantity passed in the input flist.
- **8** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the requested extension amount or quantity only).

Note: If you specify incremental mode (**8**), you can also specify whether to adjust the quantity by setting the PIN_FLD_RATING_MODE input flist field to the following:

- **0**: Calculates the reauthorization amount by adding the customer's previous reservation balance and the value passed in the input flist. This is the default.
- **1**: Calculates the reauthorization amount by adding the amount in the **/active_session** object and the value passed in the input flist.

- If you have enabled in-session notifications and the call request is not denied, the opcode returns additional in-session notifications for credit threshold breaches, service expirations, streaming usage threshold summaries, tariff change indications and subscriber preferences for such notifications.

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about the reauthorization process, see "How BRM Reauthorizes Prepaid Services".

Updating and Reauthorizing Prepaid Sessions

To update customer usage data and reauthorize a prepaid session in one transaction, call the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode with the following information in the input flist:

- Service type
- Device ID
- Direction of the call
- Name of the calling application
- Requested reauthorization amount or quantity
- Details about the call, such as the IMEI value, the dialed number, and the QoS

Note: If you are not using IMDB Cache for AAA, you must also pass the PIN_FLD_RESERVATION_OBJ field in the opcode's input flist. For more information, see "Calculating Reservation Balances for Non-IMDB Cache Systems".

You can specify whether the reauthorization amount or quantity is cumulative or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the update amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session).
- **2** specifies that the update amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the value in the **/active_session** object. This is the default update mode.
- **4** specifies that the reauthorization amount or quantity passed in the input flist is *cumulative* (that is, it represents the original authorization amount plus a requested

extension amount). BRM reauthorizes by using the amount or quantity passed in the input flist.

- **8** specifies that the reauthorization amount or quantity passed in the input flist is *incremental* (that is, it represents the requested extension amount or quantity only). This is the default reauthorization mode.

  Note: If you specify incremental mode (**8**), you can also specify whether to adjust the quantity by setting the PIN_FLD_RATING_MODE input flist field to the following:

  - **0**: Calculates the reauthorization amount by adding the customer's previous reservation balance and the value passed in the input flist. This is the default.
  - **1**: Calculates the reauthorization amount by adding the amount in the **/active_session** object and the value passed in the input flist.

- If you have enabled in-session notifications and the call request is not denied, the opcode returns additional in-session notifications for credit threshold breaches, service expirations, streaming usage threshold summaries, tariff change indications and subscriber preferences for such notifications.

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about the updating and reauthorization process, see "How BRM Updates and Reauthorizes Prepaid Sessions".

Canceling Authorization for Prepaid Services

To cancel an existing authorization and return reserved resources back to the customer's account balance, call the PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION opcode with the following information in the input flist:

- Service type
- Device ID
- Authorization ID

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about the cancellation process, see "How BRM Cancels Prepaid Service Authorizations".

Rating and Recording Activity Events

To rate and record activity events, such as purchasing a product or changing a password, call the PCM_OP_TCF_AAA_ACCOUNTING opcode with the following information in the input flist:

- Service type
- MSID
- Name of the calling program
- Authorization ID

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about rating activity events, see "How BRM Rates and Records Prepaid Activity Events".

Managing Prepaid Sessions

After a customer is authorized to use a service, the external network connects the call and begins collecting information about the customer's usage, such as the starting time, the dialed number, and the direction of the call. The network sends this information to BRM, which records the information in **/active_session** objects.

When the session ends, BRM rates any usage, closes or deletes the associated **/active_session** and **/reservation/active** objects, and records the data in **/event/session** objects in the BRM database.

You use the Services Framework AAA ACCOUNTING standard opcode to perform the following tasks:

- *Start* prepaid sessions.
- *Update* information about an existing prepaid session.
- *End* prepaid sessions.
- *Close* any open sessions when the external network shuts down abnormally or restarts.

Starting Prepaid Sessions

To start a prepaid session, call the PCM_OP_TCF_AAA_START_ACCOUNTING opcode with the following information in the input flist:

- Service type
- Device ID
- Authorization ID
- Direction of the call
- Information about the call, such as the IMEI value, dialed number, and QoS

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about starting sessions, see "How BRM Starts Prepaid Sessions".

Updating a Prepaid Session

To update information about an existing prepaid session, call the PCM_OP_TCF_AAA_UPDATE_ACCOUNTING opcode with the following information in the input flist:

- Service type
- Device ID
- Authorization ID
- Direction of the call
- Details that changed, such as the quantity or amount used during the session

This opcode can also be called at any time during a session to update any inherited data (PIN_FLD_INHERITED_INFO) fields in an active session object. Only the fields specified in the opcode's input flist are updated; all other fields in the session objects are left unchanged.

You can specify whether the usage amount or quantity is cumulative or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the update amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session). BRM applies the usage amount or quantity from the input flist.
- **2** specifies that the update amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the value in the **/active_session** object.

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about updating sessions, see "How BRM Updates Prepaid Sessions".

Ending Prepaid Sessions

To end a prepaid session when a call ends successfully, call the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode with the following information in the input flist:

- Service type
- Device ID
- Direction of the call
- Details about the call, such as the IMEI value, dialed number, and QoS

You use this opcode to perform the following actions:

- Close, cancel, or delete the active session object.
- Release or delete any associated reservation objects.
- Rate any usage.
- Record information about the session into an **/event/session** object in the BRM database.

You can specify whether the usage amount or quantity is cumulative or incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the update amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session). BRM applies the usage amount or quantity from the input flist.
- **2** specifies that the update amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the value in the **/active_session** object.

PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about ending sessions, see "How BRM Ends Prepaid Sessions".

Closing Prepaid Sessions when the External Network Shuts Down

To close all open sessions when the external network is being shut down or encounters problems, call the PCM_OP_TCF_AAA_ACCOUNTING_OFF opcode with the following information in the input flist:

- Service type
- Originating network ID (SCP name)

- (Optional) Start or end time
- (Optional) Status
- (Optional) Termination cause

Sessions with a status of STARTED or UPDATED are automatically rated before they are closed. You specify how BRM handles sessions with a CREATED status by passing the optional PIN_FLD_ACC_FLAG input flist field:

- When this flag *is* passed, CREATED sessions are *rated* before they are closed.
- When the flag *is not* passed, CREATED sessions are *cancelled*.

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about closing sessions, see "How BRM Closes Prepaid Sessions When the External Network Shuts Down".

Closing Prepaid Sessions when the External Network Restarts

To close all open prepaid sessions when the external network restarts, call the PCM_OP_TCF_AAA_ACCOUNTING_ON opcode with the following information in the input flist:

- Service type
- Originating network ID (SCP name)
- (Optional) Start time
- (Optional) Status

Sessions with a status of STARTED or UPDATED are automatically rated before they are closed. You specify how BRM handles sessions with a CREATED status by passing the optional PIN_FLD_ACC_FLAG input flist field:

- When this flag *is* passed, CREATED sessions are *rated* before they are closed.
- When the flag *is not* passed, CREATED sessions are *cancelled*.

This opcode calls the helper opcodes specified in the **/config/opcodemap/tcf** object. For detailed information about closing sessions, see "How BRM Closes Prepaid Sessions When the External Network Restarts".

Note:

- PCM_OP_TCF_AAA_ACCOUNTING_ON and PCM_OP_TCF_AAA_ACCOUNTING_OFF opcodes use the default search criteria to search the active sessions. The default search criteria is based on the PIN_FLD_TELCO_INFO.PIN_FLD_ORIGIN_NETWORK field that is supported only for the **/active_session/telco** objects. For the non-telco service types, you must add a custom helper opcode for the ACC_ON_OFF_SEARCH processing stage to define the search criteria for the non-telco active session type.
- For example, you can add a custom helper opcode for ACC_ON_OFF_SEARCH processing stage for a non-telco service type, say **/active_session/ip** object. This opcode defines the search criteria to find the list of active sessions from the same NAS. In the custom opcode, you can add name of the NAS as one of the search criteria.

Requesting an Account's Balance Information

To request an account's balance information, call the PCM_OP_TCF_AAA_QUERY_BALANCE opcode.

This opcode:

1. Calls the PCM_OP_ACT_FIND opcode to get the user's **/account** and **/service** object POIDs.
2. Calls the PCM_OP_BAL_GET_BALANCES opcode to get the user's account balance information.

Requesting Service Price Information

To request the cost of a specific service, call the PCM_OP_TCF_AAA_SERVICE_PRICE_ENQUIRY opcode.

This opcode:

1. Calls the PCM_OP_ACT_FIND opcode to get the user's **/service** and **/account** object POIDs.
2. Calls the PCM_OP_ACT_USAGE opcode to get cost of the service.

Preparing Service-Specific Data by Using Helper Opcodes

By default, Services Framework AAA Manager uses these helper opcodes to prepare service-specific data.

Important: Do not call these opcodes directly. You configure opcodes to call helper opcodes by using the "load_aaa_config_opcodemap_tcf". See "Configuring Services Framework to Call Helper Opcodes".

- To prepare input flists for authorization, use PCM_OP_TCF_AAA_AUTHORIZE_PREP_INPUT. See "Preparing Service-Specific Flists for Authorization".
- To build service-specific search templates, use PCM_OP_TCF_AAA_ACCOUNTING_PREP_INPUT. See "Building Service-Specific Search Templates".
- To generate service-specific flists for reauthorization, use PCM_OP_TCF_AAA_REAUTHORIZE_PREP_INPUT. See "Preparing Service-Specific Flists for Reauthorization".
- To generate service-specific flists for ending sessions, use PCM_OP_TCF_AAA_STOP_ACCOUNTING_PREP_INPUT. See "Preparing Service-Specific Flists for Ending Accounting Sessions".
- To generate service-specific flists for updating sessions, use PCM_OP_TCF_AAA_UPDATE_ACCOUNTING_PREP_INPUT. See "Preparing Service-Specific Flists for Updating Accounting Sessions".
- To generate service-specific flists for activity events, use PCM_OP_TCF_AAA_ACCOUNTING_PREP_INPUT. See "Preparing Service-Specific Flists for Activity Events".

Preparing Service-Specific Flists for Authorization

Use the PCM_OP_TCF_AAA_AUTHORIZE_PREP_INPUT helper opcode to aggregate service-specific data and then prepare an flist for authorizing a prepaid accounting session.

This opcode aggregates data by amount, duration, volume, or activity, depending on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies to rate the *amount*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Amount-Based Aggregation:

When aggregating the amount, the helper opcode prepares the PIN_FLD_BALANCES array in the PIN_FLD_RATING_INFO substruct, indexed by the currency type.

For Duration-Based Aggregation:

When aggregating the duration, the helper opcode performs the following:

1. Assigns a starting timestamp to the PIN_FLD_START_T field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct. The opcode uses the starting timestamp from the input flist or, if one is not passed in, from "pin_virtual_time" (see *BRM Developer's Guide*).

   Note: This is a temporary starting timestamp only and is later replaced with the actual starting timestamp by the reauthorization or stop accounting opcodes.

2. Assigns an ending timestamp to the PIN_FLD_END_T field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct:
   - If PIN_FLD_END_T is supplied in the input flist, the opcode assigns the ending timestamp directly to the PIN_FLD_END_T field.
   - If PIN_FLD_QUANTITY is supplied in the input flist, the opcode calculates the ending timestamp by adding the quantity passed in the input flist to the starting timestamp.
   - If an ending timestamp or quantity is not passed in, the opcode retrieves the default authorization quantity from the **/config/reserve** object. The opcode calculates the ending timestamp by adding the default authorization quantity to the starting timestamp.

For Volume-Based Aggregation:

When aggregating the volume, the helper opcode adds the bytes uploaded and the bytes downloaded and assigns the value to the PIN_FLD_BYTES_UPLINK and PIN_FLD_BYTES_DOWNLINK fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT.PIN_FLD_TELCO_INFO substruct.

Building Service-Specific Search Templates

Use the PCM_OP_TCF_AAA_SEARCH_SESSION helper opcode to build a search template for finding **/active_session/telco** objects or **/event/session/telco** objects. These search templates are used by the Services Framework AAA opcodes to look for duplicate session objects.

By default, this helper opcode sets the search criteria to the following, but you can customize it to use other criteria:

- Authorization ID
- APN name
- GGSN address
- SGSN address
- Starting timestamp
- For **/active_session/telco** objects, the authorization ID.
- For **/event/session/telco** objects, the network session ID, MSISDN, and start time.

Preparing Service-Specific Flists for Reauthorization

Use the PCM_OP_TCF_AAA_REAUTHORIZE_PREP_INPUT helper opcode to aggregate service-specific data and then prepare an flist for reauthorizing a prepaid accounting session.

This opcode aggregates data by amount, duration, volume, or activity, depending on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies to rate the *amount*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Amount-Based Aggregation:

When aggregating the amount, the helper opcode assigns a reauthorization amount to the PIN_FLD_AMOUNT field in the PIN_FLD_RATING_INFO substruct. The method the opcode uses to calculate the reauthorization amount depends on the value passed in the PIN_FLD_AGGREGATE_MODE field:

- When PIN_FLD_AGGREGATE_MODE is set to **4**, the opcode uses the amount passed in the input flist.
- When PIN_FLD_AGGREGATE_MODE is set to **8**, the opcode calculates the reauthorization amount based on the PIN_FLD_RATING_MODE field:
  - When PIN_FLD_RATING_MODE is **0**, the opcode adds the amount passed in the input flist to the amount in the **/active_session** object.
  - When PIN_FLD_RATING_MODE is **1**, the opcode adds the amount passed in the input flist to the amount in the **/reservation/active** object.

For Duration-Based Requests:

When aggregating the duration, the helper opcode assigns a starting timestamp and ending timestamp to the PIN_FLD_START_T and PIN_FLD_END_T fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct.

The opcode assigns a starting timestamp based on the following:

- If PIN_FLD_START_T is passed in the input flist and the timestamp is earlier than the timestamp in the **/active_session** object, the opcode assigns the starting timestamp from the input flist.
- If PIN_FLD_START_T is passed in the input flist and the **/active_session** object does not already exist, the opcode assigns the starting timestamp from the input flist.
- If PIN_FLD_START_T is not passed in and the **/active_session** object does not already exist, the opcode assigns the starting timestamp from "pin_virtual_time" (see *BRM Developer's Guide*).

The opcode assigns an ending timestamp based on the following:

- If PIN_FLD_END_T is passed in the input flist, the opcode assigns the ending timestamp from the input flist.
- If PIN_FLD_QUANTITY is passed in the input flist, the opcode calculates the ending timestamp based on the values passed in the PIN_FLD_AGGREGATE_MODE field:
  - When PIN_FLD_AGGREGATE_MODE is **4**, the opcode calculates the ending timestamp by adding the starting timestamp and the quantity.
  - When PIN_FLD_AGGREGATE_MODE is **8**, the opcode populates the value based on the PIN_FLD_RATING_MODE field:
    * When PIN_FLD_RATING_MODE is **0**, the opcode calculates the ending timestamp by adding together the ending timestamp from the **/active_session** object and the quantity from the input flist.
    * When PIN_FLD_RATING_MODE is **1**, the opcode calculates the ending timestamp by adding together the starting timestamp, the reserved quantity from **/reservation/active**, and the quantity from the input flist.
- If an ending timestamp or quantity is not passed in, the opcode retrieves the default reauthorization quantity from the **/config/reserve/gprs** object. The opcode calculates the ending timestamp by adding together the default reauthorization quantity and the starting timestamp.

For Volume-Based Requests:

When aggregating the volume, the helper opcode assigns the reauthorization volume to the PIN_FLD_BYTES_UPLINK and PIN_FLD_BYTES_DOWNLINK fields in the PIN_FLD_RATING_INFO substruct. The method the opcode uses to calculate the reauthorization volume depends on the value passed in the PIN_FLD_AGGREGATE_MODE field:

- When PIN_FLD_AGGREGATE_MODE is set to **4**, the opcode assigns the volume passed in the input flist.
- When PIN_FLD_AGGREGATE_MODE is set to **8**, the opcode calculates the reauthorization volume based on the PIN_FLD_RATING_MODE field:
  - When PIN_FLD_RATING_MODE is set to **0**, the opcode adds the bytes uploaded or downloaded from the input flist to the value in the **/active_session** object.
  - When PIN_FLD_RATING_MODE is set to **1**, the opcode adds the bytes uploaded or downloaded from the input flist to the value in the **/reservation/active** object.

Preparing Service-Specific Flists for Ending Accounting Sessions

Use the PCM_OP_TCF_AAA_STOP_ACCOUNTING_PREP_INPUT helper opcode to aggregate service-specific data and then prepare an input flist for ending a prepaid accounting session.

This opcode aggregates service-specific data based on the value passed in the PIN_FLD_SUBSESSION_MODE flist field:

- **1** specifies *aggregate mode*.
- **2** specifies *rate immediately*.
- **3** specifies deferred rate mode.

For more information about the rating modes, see "Specifying How to Rate Subsessions".

For aggregate mode:

When set to aggregate mode, the helper opcode reads the PIN_FLD_STOP_INDICATOR field to determine whether the session is still in progress (**0**) or has ended (**1**). When the field is set to **0**, the helper opcode does nothing. When the field is set to **1**, the helper opcode performs the following:

1. Aggregates the amount or quantity passed in the input flist with the amount or quantity in the **/active_session** object.
2. Aggregates the volume information from all of the subsession objects and records it in the master session object.
3. Determines the session's starting timestamp and ending timestamp by choosing the earliest and latest timestamps from all of the subsession objects and then records them in the master session object.
4. Sets the status of all subsession objects to **Closed And Unrated**, which indicates that the objects should be closed and the event should not be recorded.

If it is a master session object, the information is passed in the top level of the flist; if it is a subsession object, the information is passed in the PIN_FLD_SESSION_INFO array.

For Rate Immediately:

When set to rate immediately, the helper opcode reads the PIN_FLD_STOP_INDICATOR field to determine whether the session is still in progress (**0**) or has ended (**1**). When the field is set to **0**, the helper opcode does nothing. When the field is set to **1**, the helper opcode aggregates the amount or quantity passed in the input flist with the amount or quantity in the **/active_session** object and then passes the **/active_session** object with the status set to **Closed**. If it is a master session object, the information is passed in the top level of the flist; if it is a subsession object, the information is passed in the PIN_FLD_SESSION_INFO array.

For Deferred Rate Mode:

When set to deferred rate mode, the helper opcode reads the PIN_FLD_STOP_INDICATOR field to determine whether the session is still in progress (**0**) or has ended (**1**). When the field is set to **0**, the helper opcode does nothing. When the field is set to **1**, the helper opcode aggregates the amount or quantity passed in the input flist with the amount or quantity in the **/active_session** object and then passes all of the master and subsession objects with the status set to **Closed**, which indicates that all of the objects should be rated and recorded as events. If it is a master session object, the information is passed in the top level of the flist; if it is a subsession object, the information is passed in the PIN_FLD_SESSION_INFO array.

Preparing Service-Specific Flists for Updating Accounting Sessions

Use the PCM_OP_TCF_AAA_UPDATE_ACCOUNTING_PREP_INPUT helper opcode to aggregate data and then prepare an flist for updating an existing prepaid accounting session.

This opcode aggregates service-specific data by amount, duration, volume, or activity, depending on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies to rate the *amount*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Amount-Based Aggregation:

When aggregating the amount, the helper opcode prepares the PIN_FLD_BALANCES array in the PIN_FLD_RATING_INFO substruct, indexed by the currency type.

- If PIN_FLD_AGGREGATE_MODE is **4**, the opcode assigns the amount from the input flist.
- If PIN_FLD_AGGREGATE_MODE is **8**, the opcode calculates the amount by adding together the amount passed in the input flist and the amount from the **/active_session** object.

For Duration-Based Requests:

When aggregating the duration, the helper opcode assigns a starting timestamp and an ending timestamp to the PIN_FLD_START_T and PIN_FLD_END_T fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct.

- For the starting timestamp, the helper opcode assigns the timestamp from either the **/active_session** object or the input flist, depending on which one has the earliest start time. If a starting timestamp is not present in the input flist or the **/active_session** object, the helper opcode does not modify the existing starting timestamp.
- For the ending timestamp, the helper opcode assigns the timestamp from either the **/active_session** object or the input flist, depending on which one has the latest end time. If an ending timestamp is not present in the input flist or the **/active_session** object, the helper opcode does not modify the existing ending timestamp.

For Volume-Based Requests:

When aggregating the volume, the helper opcode assigns the volume uploaded or downloaded by the customer to the PIN_FLD_BYTES_UPLINK or PIN_FLD_BYTES_DOWNLINK field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT.PIN_FLD_TELCO_INFO substruct. The method the helper opcode uses to calculate the volume depends on the value passed in the PIN_FLD_AGGREGATE_MODE flist field:

- When PIN_FLD_AGGREGATE_MODE is **4**, the opcode uses the bytes passed in the input flist.
- When PIN_FLD_AGGREGATE_MODE is **8**, the opcode calculates the volume by adding together the bytes passed in the input flist and the bytes in the **/active_session** object.

Preparing Service-Specific Flists for Activity Events

Use the PCM_OP_TCF_AAA_ACCOUNTING_PREP_INPUT helper opcode to aggregate service-specific data and then prepare an flist for rating activity events.

The data returned by the helper depends on the information passed in the input flist:

- When **/active_session** object details are passed in the input flist, the helper opcode merges the session information from the input flist and the **/active_session** object.
- When neither PIN_FLD_START_T or PIN_FLD_END_T are passed in the input flist, the helper opcode sets the starting timestamp to "pin_virtual_time" in *BRM Developer's Guide*.
- When the amount is passed in the input flist, the helper opcode prepares the PIN_FLD_BAL_IMPACTS array, indexed by currency type.

21

Adding New Prepaid Services

This chapter describes how to add a prepaid service to your Oracle Communications Billing and Revenue Management (BRM) system by using Services Framework Manager.

About Adding a Prepaid Service

Complete these tasks to add a new prepaid service to your BRM system:

1. Extending Configuration, Event, Service, and Device Objects.
2. Creating and Loading Configuration Files.
3. Modifying Policy Files.
4. Modifying service-specific helper opcodes. See "Services Framework AAA Manager FM Helper Opcodes" in *BRM Developer's Reference*.
5. Writing custom opcodes that run during service order and device state changes. See "Writing a Custom Facilities Module" in *BRM Developer's Guide*.
6. Modifying the XML *provisioning payload* file created by the Services Provisioning Data Manager. See "load_pin_telco_provisioning".
7. Modifying Customer Center to Support Your New Service.
8. Testing Provisioning by the Network Simulator.

Extending Configuration, Event, Service, and Device Objects

This section describes how to extend objects to support your new service (*service_name*).

1. Make sure your Oracle Data Manager (**dm_oracle)** configuration file (*BRM_Home*/**sys/dm_oracle/pin.conf)** has **write_enable** parameters set to **1**:

   ```
   - dm dd_write_enable_objects 1
   - dm dd_write_enable_fields 1
   - dm dd_write_enable_portal_objects 1
   ```

2. Extend the classes listed in Table 21–1 by using Storable Class Editor.

   In this table, *service_name* indicates your service name:

Table 21–1 Class Extensions

Extension	Notes
/config/telco/*service_name*	Do not add new fields to this extended class.
/config/telco/service_order_state/*service_name*	Do not add new fields to this extended class.
/event/provisioning/service_order/telco/*service_name*	Instances of this class are the provisioning service orders for your new service. (Optional) Add fields to the extended class.
/service/telco/*service_name*	This class contains the service definitions for your new service.

3. If the new service requires a new device or devices in addition to the default number device (**/device/num**), extend the classes listed in Table 21–2 by using Developer Center:

   In this table, *device_name* indicates your device name.

Table 21–2 Device Extensions

Extended Class	Notes
/device/*device_name*	None
/config/device_state/*device_name*	See **load_pin_device_state**

For more information about creating custom fields, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

Creating and Loading Configuration Files

This section describes how to create and load configuration files to support your new service.

1. Check in the directory where you run the load utilities to verify that you have a configuration (**pin.conf**) file that connects to the correct Connection Manager (CM).
2. Create a localized string configuration file and load it by running the **load_localized_string** utility.
3. Load any new *devices*. See "Installing and Configuring Number Manager and Number Administration Center".

   Important: If you are using **/device/num**, use Number Administrator to load the devices. If you are creating a custom device, you must create and run a load utility for it.

4. Define the device-to-service mappings in a text file and load it by running the **load_pin_device_permit_map** utility.

   The default device-to-service mappings text file shipped with Number Manager is *BRM_Home***/sys/data/config/pin_device_permit_map**.

   See "Defining Device-to-Service Associations" in *BRM Developer's Guide*.

5. Add provisioning configuration information to a text file and load it by running the **load_pin_telco_provisioning** utility.

   The following sample provisioning definition is for a service called **cellular**:

   ```
   Service provisioning info:

   /service/telco/cellular, D,
   "NAME" "MOBTEL",
   PIN_FLD_TELCO_FEATURES[*].PIN_FLD_NAME
   /device/num, I, PIN_FLD_DEVICE_ID "MSISDN"
   ```

   Tip: See the sample input file *BRM_Home*/**sys/data/config/pin_conifg_telco_service_order**.

   After you load this file, a single instance of **/config/telco/***service_name* is created.

6. Define provisioning service order states in a text file and load it with the **load_pin_telco_service_order_state** utility.

   The following sample state definition is for a service called **cellular**:

   ```
   /config/telco/service_order_state/cellular
   /event/provisioning/service_order/telco/cellular
   # NEW -> READY (starting state)
   1: 1: 0: 0
   2: 0:0
   3: 0:0
   4: 0:0
   5: 0:0
   ```

   Tip: See the sample input file *BRM_Home*/**sys/data/config/pin_telco_svc_order_state**.

   After you load this file, a single instance of **/config/telco/service_order_state** is created.

7. Define provisioning tags, service features, and account- and service-level extended rating attributes (ERAs) in a text file and load it with the **load_pin_telco_tags** utility.

   The following sample provisioning tags entry is in an input file for a service called **cellular**:

   ```
   provisioning_tag       "/config/telco/cellular"       "Voice Add On Corporate
   Plus: Promotions and
   SS"    "Promotion and SS (currently: MPTY and F&F)" "y"
   features                "MPTY"
   service_era             "FRIENDS_FAMILY" 12 13 "n"
   service_era             "HOME_REGION"   14 15 "y"
   account_era             "SPECIAL_DAY"   2 3
   ```

   After you load this file, a single instance of **/config/telco/***service_name* and **/config/account-era** is created.

This completes creating and loading configuration files.

Modifying Policy Files

You may need to modify some policy opcodes to support your new service.

1. Configure these subscription policy opcodes to load and validate tags for the new service:
   - PCM_OP_SUBSCRIPTION_POL_PURCHASE_PROD_PROVISIONING
   - PCM_OP_SUBSCRIPTION_POL_CANCEL_PROD_PROVISIONING
   - PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS

   Tip: For examples, look for **tag_fn_tcf** functions and functions defined for **/service/telco**. Copy and extend these functions as required.

2. Generate the facilities module **fm_subscription_pol.so** by using the **make** command and move the file to the BRM library.
3. (Optional) Modify the PCM_OP_CUST_POL_PREP_INHERITED opcode to capture and process additional information for **/service/telco/***service_name*.
4. Generate the facilities module **fm_cust_pol.so** by using the **make** command and move the file to the BRM library.
5. (Optional) Add any required code to a policy opcode and modify PCM_OP_TCF_POL_APPLY_PARAMETER to call it. See the PCM_OP_GSM_POL_APPLY_PARAMETER code for a service-specific example.
6. Generate the facilities module **fm_tcf_pol.so** by using the **make** command and move the file to the BRM library.
7. Stop and restart Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

   Tip: Set the log level to 3 to see if all the provisioning tags and features are loaded for **/config/telco/***service_name* during startup.

8. Define a pricing plan based on the extended service.

This completes policy file modifications.

Modifying Customer Center to Support Your New Service

This section describes how to configure Customer Center to support your new service.

Before modifying Customer Center service and device panels, you should read "About Customizing the Services Framework Manager Client".

Create Service Panels

1. Create an extended service panel for any additional information needed for the service that is not already captured in the device panels. See "Creating Custom Service Panels".
2. If you are using a device other than **/device/num**, create a panel for each device that will be associated with the service. See "Creating Custom Device Panels".

3. Create a device search entry panel for each device. See "Creating Custom Device Search Panels".
4. Create a device search results panel for each device. See "Creating Custom Device Search Results Panels".

Configure Customer Center by Using Configurator

To configure Customer Center:

1. Start Configurator. See "Using Configurator to Configure Customer Center" in *BRM Developer's Guide*.
2. Specify any extended service panels and configure layout options by using Service Configurator. See "Telco Service Configurator".
3. Specify any device panels and configure layout options by using Device Configurator. See "Telco Device Configurator".

You can now create accounts for the new service using the price plan defined earlier. You can see service orders being generated by using Event Browser.

> **Note:** You do not need to modify the History and Promotion panels to support your new service. When you view the history of a device, the device type is automatically passed in. This enables you to view the history of any device associated with any service without any customization. The Promotion panel automatically displays all account- and service-level ERAs belonging to an account.

Testing Provisioning by the Network Simulator

You can test provisioning of your services by using the Network Simulator. See "Testing Provisioning Using BRM Network Simulator".

22

Testing Provisioning Using BRM Network Simulator

This chapter describes how to use the Oracle Communications Billing and Revenue Management (BRM) Network Simulator to test your Telco implementation.

About the Network Simulator

The Network Simulator simulates a network provisioning agent by receiving and processing XML provisioning payload files from the Provisioning DM.

> **Important:** To use this simulator, the Provisioning DM must be running.

The Network Simulator includes the "Perl-Based Agent" and the "Java-Based Simulator".

Perl-Based Agent

The Perl-based agent (**agent_sim.pl**) captures Service Order XML provisioning payload files from the Provisioning DM as follows:

- If the $PIN_HOME environment variable is set, the Perl-based agent captures and saves the XML file in *BRM_Home***/apps/telco/service_orders** with the name **SvcOrder**_*EventObject_id*.**xml**.
- If the $PIN_HOME environment variable is not set, the XML file is created in the directory where the simulator is launched.

Java-Based Simulator

The Java-based simulator:

1. Processes the XML file according to command-line arguments.
2. Updates the service order status by calling the PCM_OP_PROV_UPDATE_SVC_ORDER opcode.

Using the Network Simulator

Important: Before running Network Simulator, enable provisioning. See "Enabling Wireless Provisioning".

To use the Network Simulator:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Uncomment the **agent_return** entry and set it to **1**:

   ```
   - fm_tcf agent_return 1
   ```

3. Be sure that the internal simulator is turned off by setting the **simulate_agent** entry to **0**.

   ```
   - fm_tcf simulate_agent 0
   ```

4. Save the file.
5. Restart the CM.
6. Start the Perl-based agent:

   ```
   Perl agent_sim.pl &
   ```

7. Start the Provisioning DM:

   ```
   start_dm_prov_telco
   ```

   Important: Always start the Perl-based agent before you start the Provisioning DM.

8. Create a provisioning event.

   Tip: You can create a provisioning event by using Customer Center to create an account and purchase a product that requires provisioning.

9. (Optional) To view the XML file generated for the service order by the Provisioning DM:
   a. Open Event Browser.
   b. Open the service order object (**/event/provisioning/service_order/telco***EventObject_ID*) and note the event object ID.
   c. Open the associated XML file, **SvcOrder**_*EventObject_ID*.**xml**, in the *BRM_Home*/**apps/telco/service_orders** directory by using a text editor.
10. To test provisioning by simulating the network, run the Java-based simulator **RunSimulator**. See "Running the Java-Based Simulator".

Running the Java-Based Simulator

You use the Java-based simulator command **RunSimulator** to process the XML provisioning payload file that was created with the Perl-based agent.

If you include a parameter to change the state of a supplied service order, the status of associated service changes accordingly. For example, when a successful service order is returned, the service, product, and supplementary service (feature) status is set to **Active**.

To run the Java-based simulator:

- Run **RunSimulator** at the command prompt:

```
RunSimulator [-d] [-s status] [-f status_flags] [-t feature_status][-m status_msg] [-x xml_file]
```

Parameter Descriptions

-d

Debug mode.

-s

Status of service order.

Possible values: **0** (success; this is the default), **1** (failed)

-f

Status flag of service order to be returned.

-t

Supplementary service (feature) status to be returned.

-m

Status message.

-x

XML provisioning payload file.

Java-Based Simulator Sample Command

This is a sample command line for running the Java-based simulator in Network Simulator:

```
RunSimulator -x SvcOrder_EventObject_id.xml
```

23

Services Framework Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Services Framework utilities.

load_pin_network_elements

Use the **load_pin_network_elements** utility to load network elements used by SIM Manager and Number Manager into the **/config/network_element** object.

For more information, see "Installing and Configuring Number Manager and Number Administration Center" and "Installing and Configuring SIM Manager and SIM Administration Center".

Note: You cannot load separate **/config/network_element** objects for each brand. All brands use the same object.

Important: To connect to the Oracle Communications Billing and Revenue Management (BRM) database, the **load_pin_network_elements** utility needs a **pin.conf** 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_network_elements [-d] [-v] [-i|-r] pin_network_elements_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

-i
Appends all new network elements in the **pin_network_elements** file to the existing elements in the **/config/network_element** object. This is the default.

-r
Deletes the existing **/config/network_element** object and creates a new object with the network elements provided in the **pin_network_elements** file.

pin_network_elements_file
Specifies the name of the network elements file. The default file is *BRM_Home*/**sys/data/config/pin_network_elements**.

Results

If the utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was run or in a directory specified in the **pin.conf** configuration file.

To verify that the network elements were loaded, you can display the **/config/network_element** 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*.

Important: You must restart the Connection Manager (CM) to make new or changed settings available to BRM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_service_framework_permitted_service_types

Use the Oracle Communications Billing and Revenue Management (BRM) **load_pin_service_framework_permitted_service_types** utility to load the list of non-telco service types that are supported by the Services Framework into the **/config/service_framework/permitted_service_types** object in the BRM database. You define the supported non-telco service types in the *BRM_Home***/sys/data/config/pin_service_framework_permitted_service_types.xml** file.

For more information, see "Specifying the Non-Telco Services Supported by Services Framework".

Caution: This utility overwrites pre-existing data in the **/config/service_framework/permitted_service_types** object. If you are adding new information, you must also include the pre-existing data in the **pin_service_framework_permitted_service_types.xml** file.

Important: To connect to the BRM database, the **load_pin_service_framework_permitted_service_types** utility needs a **pin.conf** 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_service_framework_permitted_service_types [-f xml_file]
                                                    [-v] [-h]
```

Parameters

-f xml_file
The name and location of the XML file that defines the list of non-telco service types that are supported by Services Framework. By default, the utility uses the *BRM_Home***/sys/data/config/pin_service_framework_permitted_service_types.xml** file.

If you do not run the utility from the directory in which the file is located, you must include the complete path to the file.

-v
Displays information about successful or failed processing as the utility runs.

-h
Displays the syntax and parameters for this utility.

Results

The utility notifies you when it successfully creates the **/config/service_framework/permitted_service_types** object or if it encountered errors. You can view more detailed error messages by looking in the **default.pinlog** file. This file is either in

the directory from which the utility was started or in a directory specified in the utility configuration file.

load_pin_telco_provisioning

Use this utility to load your provisioning configuration into the Oracle Communications Billing and Revenue Management (BRM) database.

The provisioning configuration specifies the service, device, and profile object fields that need to be added to a provisioning service order. You specify the configuration in a provisioning configuration file (*telco_prov_config_filename*) and use that file as input for the utility. The default file is *BRM_Home*/**sys/data/config/pin_telco_provisioning**.

This utility creates the **/config/telco/provisioning** and **/config/telco/provisioning/fieldlist** configuration objects.

For information, see "Specifying the Details to Add to the Service Order".

Note: You cannot load separate **/config/telco/provisioning** objects for each brand. All brands use the same object.

Important: To connect to the BRM database, the **load_pin_telco_provisioning** 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: This utility overwrites preexisting data in the **/config/telco/provisioning** object. If you are adding new field information, you must also include preexisting field information in the input file.

Location

BRM_Home/**bin**

Syntax

```
load_pin_telco_provisioning [-d] [-v] telco_prov_config_filename
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no displayed errors, but mapping between events and opcode numbers do not appear to be loaded correctly.

-v
Displays information about successful or failed processing as the utility runs.

telco_prov_config_filename
The name and location of the *telco_prov_config_filename* file.

If you copy *telco_prov_config_filename* to the same directory from which you run the **load_pin_telco_provisioning** 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 *telco_prov_config_filename* file is located, you must include the entire path for the file.

Results

By default, **load_pin_telco_provisioning** notifies you only if it encounters errors. However, if you use the **-v** flag, the utility displays confirmation messages when running.

Important: After you run this utility, you must restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_telco_service_order_state

Use this utility to load service order state configuration data for a specific telco service, such as GSM, into the Oracle Communications Billing and Revenue Management (BRM) database.

This utility creates the **/config/telco/service_order_state** configuration object.

Tip: Use the instructions and examples included in *BRM_Home*/**sys/data/config/pin_service_order_state** file. For more information on BRM configuration files, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

For information, see "Specifying the Available States for Each Service Order".

Note: You cannot load separate **/config/telco/service_order_state** objects for each brand. All brands use the same object.

Important:

- To connect to the BRM database, the **load_pin_telco_service_order_state** utility needs a configuration file (*service_order_state_config_file*) in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.
- Specify state configuration information for only one service in the configuration file (*service_order_state_config_file*) for this utility. Load a separate configuration file for each service.

Caution: This utility overwrites pre-existing data in the **/config/telco/service_order_state** object for a particular service.

Location

BRM_Home/**bin**

Syntax

```
load_pin_service_order_state [-d] [-v] service_order_state_config_file
```

Tip: Use a naming convention to identify the input files for each service, as in this example:

```
load_pin_service_order_state -d -v pin_telco_service_order_state_
gsm
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no displayed errors, but mapping between events and opcode numbers do not appear to be loaded correctly.

-v
Displays information about successful or failed processing as the utility runs.

service_order_state_config_file
The path and name of the service order state file that you are loading for a particular service.

If you copy *service_order_state_config_file* to the same directory from which you run the **load_pin_service_order_state** utility, you do not have to specify the path.

If you run the command in a different directory from where the *service_order_state_config_file* file is located, you must include the entire path for the file.

Results

load_pin_telco_service_order_state notifies you only if it encounters errors. However, if you use the **-v** flag, the utility displays confirmation messages when running.

Important: After you run this utility, you must restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_telco_tags

Use the **load_pin_telco_tags** utility to load provisioning tags for telco services into the appropriate /config/telco/service object and account-level extended rating attributes (ERAs) into the **/config/account_era** object in the Oracle Communications Billing and Revenue Management (BRM) database.

You define telco provisioning tags and account-level ERAs in a **pin_telco_tags**_*service* file: for example, **pin_telco_tags_gsm**: located in *BRM_Home***/sys/data/config**.

For more information, see "About Provisioning Tags for Telco Services", "Defining Provisioning Tags for Telco Services by Using the pin_telco_tags file", and "Defining Account-Level ERAs in the pin_telco_tags File".

Note: You cannot load separate **/config/account_era** objects for each brand. All brands use the same object.

Important: To connect to the BRM database, the **load_pin_telco_tags** 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*.

Caution: By default, the **load_pin_telco_tags** utility appends telco provisioning tags and account-level ERAs to the BRM database. But if you use the **-x** parameter, this utility overwrites existing telco provisioning tags and account-level ERAs. Do not use the **-x** parameter unless you are certain you want to overwrite existing objects.

Location

*BRM_Home***/bin**

Syntax

```
load_pin_telco_tags [-d] [-v] [-h] [-x] pin_telco_tags_file
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no displayed errors, but mapping between events and opcode numbers do not appear to be loaded correctly.

-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_telco_tags** *any_other_parameter* **–v >** *filename*.**log**

-h
Displays the syntax and parameters for this utility.

-x
Overwrites all account-level ERAs and telco provisioning tags in the database, including the provisioning tags created using Pricing Center, with those that are in the file.

> **Caution:** Use the **-x** parameter with care.

pin_telco_tags_file
The name and location of the **pin_telco_tags** file that you are loading. The default **pin_telco_tags** files are located in *BRM_Home*/**sys/data/config**.

If you do not run the utility from the directory in which the file is located, you must include the complete path to the file, for example:

```
load_pin_telco_tags BRM_Home/sys/data/config/pin_telco_tags_gsm
```

> **Tip:** If you copy the **pin_telco_tags** file to the directory from which you run the **load_pin_telco_tags** utility, you do not have to specify the path. If you do not include a file name, the utility loads the default generic telco file, **pin_telco_tags**.

Results

The **load_pin_telco_tags** utility notifies you only if it encounters errors. However, if you use the **-v** parameter, the utility displays confirmation messages when running.

To verify that the objects were loaded, you can display the **/config/telco**_*service* object (for example, **/config/telco/gsm**) and the **/config/account_era** object by using the Object Browser, or use the **robj** command with the **testnap** utility. See "Reading an Object and Fields" in *BRM Developer's Guide*.

> **Important:** You must restart the Connection Manager (CM) to make new provisioning tags and ERAs available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

24

Services Framework AAA Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Services Framework AAA utilities.

load_config_reservation_aaa_prefs

Use this utility to specify the default authorization and reauthorization values for a telco service type. This utility loads information from an input file into a service-specific **/config/reserve** object in the Oracle Communications Billing and Revenue Management (BRM) database.

For more information about using the **load_config_reservation_aaa_prefs** utility, see "Specifying default authorization and reauthorization values".

Note: This utility is brand-aware. You can load separate **/config/reserve/gprs** objects for each brand.

Important: To connect to the BRM database, this utility needs a configuration file in the directory from which it is run. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_Home***/bin**

Syntax

```
load_config_reservation_aaa_prefs [-d] [-v] [-h] reservation_prefs
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the reservation preferences have not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used 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_config_reservation_aaa_prefs *any_other_parameter* **–v >** *filename***.log**

-h
Displays help information for using this utility.

reservation_prefs
The name and location of the file that defines your preferences. The default input file is *BRM_Home*/**sys/data/config/pin_config_reservation_aaa_prefs**.

If you run the command in a different directory from where the reservation_prefs file is located, you must include the entire path for the file.

Results

The **load_config_reservation_aaa_prefs** utility returns the following message if the **/config/reserve** object is successfully updated:

```
load_config_reservation_aaa_prefs: Successfully updated the object
/config/reserve.
```

The utility returns an error if:

- The specified input file does not exist or cannot be parsed by the utility.
- The utility cannot connect to the BRM server or write to the database.
- Any of the field names in the input file are not in the BRM system.

load_aaa_config_opcodemap_tcf

Use this utility to:

- Load new or updated opcode mappings from a text file into the **/config/opcodemap/tcf** object in the Oracle Communications Billing and Revenue Management (BRM) database.
- Export opcode mappings from the **/config/opcodemap/tcf** object into a text file.

For more information, see "Configuring Services Framework to Call Helper Opcodes".

Note: You cannot load separate **/config/opcodemap/tcf** objects for each brand. All brands use the same object.

Important: To connect to the BRM database, this utility requires a configuration file in the directory from which you run it. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Location

BRM_Home/**bin**

Syntax

```
load_aaa_config_opcodemap_tcf  -i | -f | -r [pin_config_opcodemap_tcf] [-d] [-v]
```

Parameters

-i
Specifies to append information from the opcodemap file to the existing **/config/opcodemap/tcf** object.

-f
Specifies to overwrite the existing **/config/opcodemap/tcf** object with information from the opcodemap file.

-r
Specifies to export data from the **/config/opcodemap/tcf** object into a text file.

pin_config_opcodemap_tcf
Specifies the name and location of the opcodemap file. The default opcodemap files are:

- For importing: *BRM_Home*/**sys/data/config/pin_config_opcodemap_tcf**.
- For exporting: *BRM_Home*/**sys/data/config/pin_config_opcodemap_tcf.out**

-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 file has not been loaded into the database.

-v

Displays information about successful or failed processing as the utility runs.

> **Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename*.**log** with the name of the log file:
>
> **load_aaa_config_opcodemap_tcf** *any_other_parameter* **–v >** *filename*.**log**

Error Types

This utility returns an error if:

- You use more than one of the three usage options.
- When using the **-f** or **-i** option:
  - The specified input file does not exist.
  - The utility cannot parse the input file.
- When using the **-r** option, the specified export file already exists.
- The utility cannot connect to the BRM server or write to the database.
- Any opcodes specified in the input file are not in the BRM system.

Results

The **load_aaa_config_opcodemap_tcf** utility notifies you only if it encounters errors.

load_pin_telco_aaa_params

Use the **load_pin_telco_aaa_params** utility to load configuration information for prepaid sessions into the Oracle Communications Billing and Revenue Management (BRM) database's **/config/aaa** object. You define the parameters in the *BRM_Home*/**sys/data/config/pin_telco_aaa_params.xml** file.

For more information, see "Configuring How Services Framework AAA Manages Session Objects".

Note: This utility is brand-aware. You can load separate **/config/aaa** objects for each brand.

Caution: The **load_pin_telco_aaa_params** utility overwrites existing data. If you are updating the session parameters, you cannot load new data only. You must load complete sets of data each time you run the utility.

Important: To connect to the Oracle Communications Billing and Revenue Management (BRM) database, the **load_pin_telco_aaa_params** utility needs a **pin.conf** 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_telco_aaa_params [-d] [-v] [-f pin_telco_aaa_params_file]
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

-f pin_telco_aaa_params_file
The name and location of the file that defines how to manage session objects. The default **pin_telco_aaa_params.xml** file is in *BRM_Home*/**sys/data/config**.

If you do not run the utility from the directory in which the file is located, you must include the complete path to the file, for example:

```
load_pin_telco_aaa_params -f BRM_Home/sys/data/config/pin_telco_aaa_params.xml
```

Tip: If you copy the **pin_telco_aaa_params.xml** file to the directory from which you run the **load_pin_telco_aaa_params** utility, you do not need to specify the path or file name. The file must be named **pin_telco_aaa_params.xml**.

Results

If the utility does not notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was run, or in a directory specified in the **pin.conf** configuration file.

To verify that the network elements were loaded, you can display the **/config/aaa** 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 (CM) to make new or changed settings available to BRM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

load_pin_config_auth_reauth_info

Use the **load_pin_config_auth_reauth_info** utility to load authorization and reauthorization threshold values for lightweight authorization into the **/config/auth_reauth_info** object in the Oracle Communications Billing and Revenue Management (BRM) database. You define the threshold data in the **pin_config_auth_reauth_info.xml** file in *BRM_Home*/**sys/data/config**.

For information on lightweight authorization, see "Using Lightweight Authorization".

Note: You cannot load separate **/config/auth_reauth_info** objects for each brand. All brands use the same object.

Caution: The **load_pin_config_auth_reauth_info** utility overwrites existing authorization and reauthorization threshold values. If you are updating threshold data, you cannot load new threshold data only. You must load complete sets of threshold data each time you run the **load_pin_config_auth_reauth_info** utility.

Important:

- The BRM database must be up and running.
- To connect to the BRM database, the **load_pin_config_auth_reauth_info** 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*.

Note: You can run this utility to configure authorization and reauthorization threshold values for different brands. The utility loads the **/config/auth_reauth_info** object to the current brand in which it logs in. For information on running utilities with a branded database, see "Configuring a Branded Database" in *BRM Managing Customers*.

Location

BRM_Home/**sys/data/config**

Syntax

```
load_pin_config_auth_reauth_info [-h]| [-d] [-v]| [-t] threshold_value_file_name
```

Parameters

-h
Displays online help about the command.

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

-t
Checks the validity of the XML file but does not process any data.

threshold_value_file_name
The name and location of the file that defines authorization and reauthorization threshold values for lightweight authorization. The default **pin_config_auth_reauth_info.xml** file is in *BRM_Home***/sys/data/config**.

Important: If you do not run the utility from the directory in which the file is located, you must include the complete path to the file; for example:

load_pin_config_auth_reauth_info *BRM_Home***/sys/data/pin_config_auth_reauth_info.xml**

Note: If you copy the **config_auth_reauth_info.xml** file to the directory from which you run the **load_pin_config_auth_reauth_info** utility, you do not have to specify the path or file name. By default, the file is named **pin_config_auth_reauth_info.xml**. You can change this name.

Results

The **load_pin_config_auth_reauth_info** utility notifies you when it successfully creates the **/config/auth_reauth_info** object. If it does not notify you, 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 authorization and reauthorization threshold data was loaded, you can display the **/config/auth_reauth_info** object by using Object Browser, or using 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 (CM) to make new threshold values available. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Part V

Managing GPRS Services

Part V describes how to manage GPRS services in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

25

About Performing AAA for Prepaid GPRS Services

This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) GPRS AAA Manager and describes how to implement AAA functionality in your BRM system.

Before you read this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

About Processing AAA Requests for GPRS Services

GPRS AAA Manager allows you to perform authentication, authorization, and accounting (AAA) for prepaid GPRS services. For example, when a prepaid customer accesses a GPRS service, you can use GPRS AAA Manager to perform the following:

- Verify the customer's identity by using the MSID.
- Determine whether the customer's account balance has enough resources to cover the cost of usage.
- Reserve a portion of the customer's resources for the GPRS session.
- Record usage information about the GPRS session while it is in progress.
- When the session ends, rate any usage and update the customer's account balance.

For more information about how BRM performs prepaid AAA, see "Understanding Prepaid AAA".

About GPRS AAA Manager

GPRS AAA Manager is an API that consists of opcodes, storable classes, and utilities that allow you to quickly implement AAA support for GPRS services.

About the GPRS AAA Manager Opcodes

GPRS AAA Manager includes two types of opcodes:

- **Services Framework AAA standard opcodes**. Services Framework AAA standard opcodes pass AAA requests to the BRM PCM API. These opcodes are abstract opcodes for processing AAA requests for any prepaid service type. Because the Services Framework AAA opcodes are abstract, they cannot perform GPRS-specific operations by themselves. They call helper opcodes to do this. See "About Performing AAA for Prepaid Services".

- **GPRS AAA helper opcodes**. The GPRS AAA helper opcodes perform GPRS-specific operations, such as building search templates for GPRS session objects or preparing GPRS-specific flists, for the Services Framework AAA standard opcodes. See "Preparing GPRS-Specific Data by Using Helper Opcodes".

A Services Framework AAA opcode can call a GPRS AAA helper opcode at any of these processing stages in the opcode's execution: SEARCH_SESSION, PREP_INPUT, VALIDATE_LIFECYCLE, and ACC_ON_OFF_SEARCH. The Services Framework AAA opcode determines which helper opcode to call at each processing stage by reading the **/config/opcodemap/tcf** object.

For example, at the PREP_INPUT processing stage, the Services Framework AAA opcode calls the GPRS AAA helper opcode specified in the **/config/opcodemap/tcf** object. The GPRS AAA helper opcode aggregates the GPRS data and then returns a GPRS-specific flist to the Services Framework AAA opcode. The Services Framework AAA opcode passes the flist to the PCM API, which processes the request and then returns that the request either passed or failed.

For more information, see "Services Framework AAA Manager Process Overview".

By default, the Services Framework AAA opcodes call GPRS AAA helper opcodes when processing **/service/telco/gprs** services only. You can add support for additional service types or change which helper opcodes are called by using the "load_aaa_config_opcodemap_tcf" utility. See "Configuring Services Framework to Call Helper Opcodes".

About the GPRS AAA Manager Storable Classes

By default, BRM stores information about prepaid GPRS sessions in these storable classes:

- **/active_session/telco/gprs**: Stores information about a GPRS session while it is *in progress*. This object can be subclassed for specific GPRS services.
- **/active_session/telco/gprs/master**: Stores information about a GPRS master session that is in progress.
- **/active_session/telco/gprs/master/subsession**: Stores information about a GPRS subsession that is in progress.
- **/event/session/telco/gprs**: Stores information about a GPRS session that has been *rated and closed*. This object can be subclassed for specific GPRS services.
- **/event/session/telco/gprs/master**: Stores information about a GPRS master session that has been rated and closed.

> **Note:** By default, BRM stores information about master sessions in **/active_session/telco/gprs** objects. However, to differentiate master and subsession types, you can configure BRM to store master sessions in **/active_session/telco/gprs/master** objects.

- **/event/session/telco/gprs/subsession**: Stores information about a GPRS subsession that has been rated and closed.
- **/reservation/active**: Stores information about a single reservation for one balance group.
- **/reservation_list**: Tracks the total resources a balance group has reserved in **/reservation/active** objects.

- **/reservation**: For policy-driven charging sessions, this object holds the consumed reservation amount for those resources.
- **/config/reserve/gprs**: Stores the default authorization and reauthorization values for GPRS services. This object can be subclassed for specific GPRS services.
- **/config/aaa/gprs**: Stores default preferences for processing prepaid GPRS sessions.

About the GPRS AAA Manager Utilities

GPRS AAA Manager utilities specify default preferences for processing GPRS services, including the following:

- How to store and rate GPRS accounting subsessions. See "About Tracking Data in Master Sessions and Subsessions".
- Whether to keep or delete **/active_session** and **/reservation/active** objects when a prepaid GPRS session ends.
- Whether to check for duplicate **/active_session** or **/event/session** objects.
- Default authorization and reauthorization values for GPRS services.
- The expiration time interval for **/active_session** objects stored in memory.

See "Specifying Default AAA Preferences for GPRS Services".

Setting Up Your System to Perform AAA for Prepaid GPRS Services

To set up your system to process AAA requests for prepaid GPRS services, perform the following tasks:

1. Specify your default preferences for prepaid GPRS services. See "Specifying Default AAA Preferences for GPRS Services".
2. Configure your client application to send GPRS AAA requests to the GPRS AAA Manager opcodes. See "Sending AAA Requests to GPRS AAA Manager".
3. (Optional) Customize the GPRS data used by the helper opcodes. See "Preparing GPRS-Specific Data by Using Helper Opcodes".
4. (Optional) Modify which helper opcodes are called by the Services Framework AAA opcodes or the service types that are supported. See "Configuring Services Framework to Call Helper Opcodes".

   Important: By default, the Services Framework AAA opcodes call GPRS AAA helper opcodes when processing **/service/telco/gprs** events only. To call the helper opcodes when processing additional GPRS service types, you must configure Services Framework to do so.

5. (Optional) Configure BRM to reserve a portion of a customer's resources for a prepaid GPRS session by installing and configuring Resource Reservation Manager. See "Reserving Resources for Concurrent Network Sessions" in *BRM Configuring and Collecting Payments*.

Specifying Default AAA Preferences for GPRS Services

You specify how BRM processes AAA requests for GPRS services by using the Services Framework AAA Manager utilities and configuration files:

- To specify the default authorization and reauthorization values for GPRS services, see "Specifying Default Authorization and Reauthorization Values".
- To specify how to handle GPRS session objects, see "Configuring How Services Framework AAA Manages Session Objects".
- To specify the service-specific helper opcodes to call, see "Configuring Services Framework to Call Helper Opcodes".
- To specify how to calculate the total resources reserved by an account, see "Configuring How BRM Calculates Reservation Balances".

Sending AAA Requests to GPRS AAA Manager

To perform AAA for prepaid GPRS services, your system must be designed to collect the information needed from the customer and pass the appropriate fields in the input flist to the Services Framework AAA opcodes.

You can use the Services Framework AAA opcodes to perform the following:

- Authorize customers to access GPRS services. See "Authorizing GPRS Services".
- Reauthorize customers to continue an existing GPRS session. See "Reauthorizing GPRS Sessions".
- Reauthorize GPRS sessions based on a customer's current usage. See "Updating and Reauthorizing GPRS Sessions".
- Cancel existing authorizations to use GPRS services. See "Canceling Authorization for GPRS Services".
- Rate and record one-time purchases or activity. See "Rating and Recording Activity Events".
- Manage GPRS sessions while they are in progress. See "Managing Prepaid GPRS Sessions".
- Customize GPRS authorization IDs. See "Customizing GPRS Authorization IDs".

Authorizing GPRS Services

To authorize prepaid customers to use GPRS services, call the PCM_OP_TCF_AAA_AUTHORIZE opcode with the following information in the input flist:

- GPRS service type
- MSID
- Name of the calling program
- GGSN address
- SGSN address
- APN name
- Starting time

You can specify what to rate by passing the optional PIN_FLD_REQ_MODE input flist field set to one of the following:

- **1** specifies that the events were *prerated*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.

- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For more information, see "How BRM Authorizes Users to Access Prepaid Services".

By default, this opcode calls the following GPRS AAA helper opcodes in Table 25–1 when processing **/service/telco/gprs** events:

Table 25–1 GPRS AAA Helper Codes

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GPRS_AAA_POL_AUTHORIZE_PREP_INPUT

Note: To configure PCM_OP_TCF_AAA_AUTHORIZE to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Reauthorizing GPRS Sessions

To reauthorize a customer to continue a GPRS session, call the PCM_OP_TCF_AAA_REAUTHORIZE opcode with the following information in the input flist:

- GPRS service type
- MSID
- Authorization ID or, if one is not provided, the GGSN address, SGSN address, APN name, and starting time
- Name of the calling program
- Requested reauthorization amount or quantity

You can specify what to rate by passing the optional PIN_FLD_REQ_MODE input flist field set to one of the following:

- **1** specifies that the events were *prerated*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

You can also specify whether the reauthorization amount or quantity has been aggregated or is incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field set to one of the following:

- **4** specifies that the reauthorization amount or quantity passed in the input flist is *aggregated* (that is, it represents the original authorization amount plus a requested extension amount). BRM reauthorizes by using the amount or quantity passed in the input flist.
- **8** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the requested extension amount or quantity only). This is the default.

Note: If you specify incremental mode (**8**), you can also specify how to calculate the reauthorization amount or quantity by setting the PIN_FLD_RATING_MODE input flist field to the following:

- **0**: Calculates the reauthorization amount or quantity by adding the value from the **/reservation/active** object to the value passed in the input flist. This is the default.
- **1**: Calculates the reauthorization amount or quantity by adding the value from the **/active_session** object to the value passed in the input flist.

For more information about this opcode, see "How BRM Reauthorizes Prepaid Services".

By default, PCM_OP_TCF_AAA_REAUTHORIZE calls the following helper opcodes in Table 25–2 when processing **/service/telco/gprs** events:

Table 25–2 AAA Reauthorize Helper Opcodes

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GPRS_AAA_POL_REAUTHORIZE_PREP_INPUT

Note: To configure PCM_OP_TCF_AAA_REAUTHORIZE to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Updating and Reauthorizing GPRS Sessions

To update customer usage data and reauthorize a prepaid GPRS session, call the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode with the following information in the input flist:

- GPRS service type
- MSID
- Authorization ID or, if one is not provided, the GGSN address, SGSN address, APN name, and starting time
- Name of the calling program
- Consumed quantity or amount
- Requested reauthorization quantity or amount

You can specify what to rate by passing the optional PIN_FLD_REQ_MODE input flist field set to one of the following:

- **1** specifies that the events were *prerated*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.

- **8** specifies to rate the *occurrence*. This applies to activity events only.

You can also specify whether the reauthorization amount or quantity has been aggregated or is incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field set to one of the following:

- **1** specifies that the update amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session).
- **2** specifies that the update amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the value in the **/active_session** object. This is the default update mode.
- **4** specifies that the reauthorization amount or quantity passed in the input flist is *aggregated* (that is, it represents the original authorization amount plus a requested extension amount). BRM reauthorizes by using the amount or quantity passed in the input flist.
- **8** specifies that the reauthorization amount or quantity passed in the input flist is *incremental* (that is, it represents the requested extension amount or quantity only). This is the default reauthorization mode.

Note: If you specify incremental mode (**8**), you can also specify how to calculate the reauthorization amount or quantity by setting the PIN_FLD_RATING_MODE input flist field to the following:

- **0**: Calculates the reauthorization amount or quantity by adding the value from the **/reservation/active** object to the value passed in the input flist. This is the default.
- **1**: Calculates the reauthorization amount or quantity by adding the value from the **/active_session** object to the value passed in the input flist.

For more information about this opcode, see "How BRM Updates and Reauthorizes Prepaid Sessions".

By default, PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE calls the following helper opcodes in Table 25–3 when processing **/service/telco/gprs** events:

Table 25–3 AAA Update and Reauthorize Helper Opcodes

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GPRS_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT and then PCM_OP_GPRS_AAA_POL_REAUTHORIZE_PREP_INPUT

Note: To configure PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Canceling Authorization for GPRS Services

To cancel an existing authorization and return reserved resources back to the customer's account balance, call the PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION opcode with the following information in the input flist:

- GPRS service type
- MSID
- Name of the calling program
- Authorization ID or, if one is not provided, the GGSN address, SGSN address, APN name, and starting time

For more information about this opcode, see "How BRM Cancels Prepaid Service Authorizations".

By default, PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION calls the following helper opcode in Table 25–4 when processing **/service/telco/gprs** events:

Table 25–4 AAA Cancel Authorization Helper Opcode

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION

Note: To configure PCM_OP_TCF_AAA_CANCEL_AUTHORIZATION to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Rating and Recording Activity Events

To rate and record activity events or other events that occur at a single point in time, such as sending an SMS message or changing a password, call the PCM_OP_TCF_AAA_ACCOUNTING opcode with the following information in the input flist:

- GPRS service type
- MSID
- Name of the calling program
- Authorization ID or, if one is not provided, the GGSN address, SGSN address, APN name, and starting time

For more information, see "How BRM Rates and Records Prepaid Activity Events".

By default, PCM_OP_TCF_AAA_ACCOUNTING calls the following helper opcodes in Table 25–5 when processing **/service/telco/gprs** events:

Table 25–5 AAA Accounting Helper Opcodes

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GPRS_AAA_POL_STOP_ACCOUNTING_PREP_INPUT

Note: To configure PCM_OP_TCF_AAA_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Managing Prepaid GPRS Sessions

After a customer is authorized to access a GPRS service, the external network connects the service and begins collecting information about the customer's usage, such as the starting time. The network sends this information to BRM, which records the information in **/active_session** objects.

When the session ends, BRM rates any usage, closes or deletes the associated **/active_session** object, and records the data in an **/event/session** object in the BRM database.

You use the AAA ACCOUNTING standard opcodes to perform the following tasks:

- *Start* a prepaid GPRS session.
- *Update* information about a prepaid GPRS session that is currently in progress.
- *End* a prepaid GPRS session.
- *Close* any open GPRS sessions when the external network shuts down abnormally or restarts.

Starting Prepaid GPRS Sessions

To start a prepaid GPRS session, call the PCM_OP_TCF_AAA_START_ACCOUNTING opcode with the following information in the input flist:

- GPRS service type
- MSID
- Authorization ID or, if one is not provided, the GGSN address, SGSN address, APN name, and starting time
- Session start time
- Name of the calling program

For more information about this opcode, see "How BRM Starts Prepaid Sessions".

By default, PCM_OP_TCF_AAA_START_ACCOUNTING calls the following helper opcode in Table 25–6 when processing **/service/telco/gprs** events:

Table 25–6 AAA Start Accounting Helper Opcode

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION

Note: To configure PCM_OP_TCF_AAA_START_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Updating a Prepaid GPRS Session

To update information about an existing prepaid GPRS session, call the PCM_OP_TCF_AAA_UPDATE_ACCOUNTING opcode with the following information in the input flist:

- GPRS service type
- MSID
- Name of the calling program
- Authorization ID or, if one is not provided, the GGSN address, SGSN address, APN name, and starting time
- Session end time
- Details that changed

You can specify whether the usage amount or quantity has been aggregated or is incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field:

- **1** specifies that the update amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session). BRM applies the usage amount or quantity from the input flist.
- **2** specifies that the update amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the value in the **/active_session** object.

For more information about this opcode, see "How BRM Updates Prepaid Sessions".

By default, PCM_OP_TCF_AAA_UPDATE_ACCOUNTING calls the following helper opcode in Table 25–7 when processing **/service/telco/gprs** events:

Table 25–7 AAA Update Accounting Helper Opcode

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GPRS_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT

Note: To configure PCM_OP_TCF_AAA_UPDATE_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Ending Prepaid GPRS Sessions

To end a prepaid GPRS session when it completes successfully, call the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode with the following information in the input flist:

- GPRS service type
- MSID
- Name of the calling program
- The amount or quantity consumed during the session

You use this opcode to perform the following operations:

- Close, cancel, or delete the **/active_session** object.
- Release or delete any associated reservation objects.
- Rate any usage.
- Record information about the GPRS session in an **/event/session** object in the BRM database.

You can specify whether the amount or quantity consumed has been aggregated or is incremental by passing the optional PIN_FLD_AGGREGATE_MODE input flist field set to one of the following:

- **1** specifies that the amount or quantity passed in the input flist is *aggregated* (that is, it represents the total amount or quantity used during the session).
- **2** specifies that the amount or quantity passed in the input flist is *incremental* (that is, it represents the amount or quantity used since BRM last updated the **/active_session** object). BRM calculates the total usage amount or quantity by adding the value passed in the input flist to the **/active_session** object's previous usage amount or quantity.

For more information about this opcode, see "How BRM Ends Prepaid Sessions".

By default, PCM_OP_TCF_AAA_STOP_ACCOUNTING calls the following helper opcodes in Table 25–8 when processing **/service/telco/gprs** events:

Table 25–8 AAA Stop Accounting Helper Opcodes

Processing Stage	Helper Opcode Called
SEARCH_SESSION	PCM_OP_GPRS_AAA_POL_SEARCH_SESSION
PREP_INPUT	PCM_OP_GPRS_AAA_POL_STOP_ACCOUNTING_PREP_INPUT

Note: To configure PCM_OP_TCF_AAA_STOP_ACCOUNTING to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Closing Prepaid GPRS Sessions when the External Network Shuts Down

To close all open GPRS sessions when the external network is being shut down or encounters problems, call the PCM_OP_TCF_AAA_ACCOUNTING_OFF opcode with the following information in the input flist:

- GPRS service type
- Originating network ID (SCP name)
- Name of the calling program
- (Optional) Start time and end time
- (Optional) Termination cause

This opcode closes all sessions that match the criteria passed in the input flist.

GPRS sessions with a status of STARTED or UPDATED are automatically rated before they are closed. You specify how BRM handles GPRS sessions with a CREATED status by passing the optional PIN_FLD_ACC_FLAG flist field:

- If this flag is passed, CREATED sessions are *rated* before they are closed.

- If the flag is not passed, CREATED sessions are *canceled*.

For more information about this opcode, see "How BRM Closes Prepaid Sessions When the External Network Shuts Down".

By default, PCM_OP_TCF_AAA_ACCOUNTING_OFF calls the following helper opcode in Table 25–9 when processing **/service/telco/gprs** events:

Table 25–9 AAA Accounting Off Helper Opcode

Processing Stage	Helper Opcode Called
ACC_ON_OFF_SEARCH	PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH

Note: To configure PCM_OP_TCF_AAA_ACCOUNTING_OFF to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Closing Prepaid GPRS Sessions when the External Network Restarts

To close all open GPRS sessions when the external network restarts, call the PCM_OP_TCF_AAA_ACCOUNTING_ON opcode with the following information in the input flist:

- GPRS service type
- Originating network ID (SCP name)
- Name of the calling program
- (Optional) Start time and end time
- (Optional) Termination cause

This opcode closes all sessions that match the criteria passed in the input flist.

GPRS sessions with a status of STARTED or UPDATED are automatically rated before they are closed. You specify how BRM handles GPRS sessions with a CREATED status by passing the optional PIN_FLD_ACC_FLAG input flist field:

- If this flag is passed, CREATED sessions are *rated* before they are closed.
- If the flag is not passed, CREATED sessions are *canceled*.

For more information about this opcode, see "How BRM Closes Prepaid Sessions When the External Network Restarts".

By default, PCM_OP_TCF_AAA_ACCOUNTING_ON calls the following helper opcode in Table 25–10 when processing **/service/telco/gprs** events:

Table 25–10 AAA Accounting On Helper Opcode

Processing Stage	Helper Opcode Called
ACC_ON_OFF_SEARCH	PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH

Note: To configure PCM_OP_TCF_AAA_ACCOUNTING_ON to call these helper opcodes when processing additional service types or to change which helper opcodes are called, see "Configuring Services Framework to Call Helper Opcodes".

Customizing GPRS Authorization IDs

Use the PCM_OP_GPRS_AAA_POL_AUTHORIZE policy opcode to generate a unique authorization ID if one is not passed in the input flist. By default, this opcode generates IDs that use the following format:

```
APN - GGSN_Address - SGSN_Address - START_T
```

However, you can customize this opcode to use another ID format.

This policy opcode is called by the PCM_OP_GPRS_AAA_POL_SEARCH_SESSION policy opcode during the authorization process.

Preparing GPRS-Specific Data by Using Helper Opcodes

Use these GPRS AAA helper opcodes to prepare GPRS-specific data for the Services Framework AAA opcodes.

Important: Do not call these helper opcodes directly. You configure an opcode to call helper opcodes by using the "load_aaa_config_opcodemap_tcf" utility. See "Configuring Services Framework to Call Helper Opcodes".

- To prepare an input flist for *authorization*, use the PCM_OP_GPRS_AAA_POL_AUTHORIZE_PREP_INPUT helper opcode. See "Preparing GPRS-Specific Flists for Authorization".
- To prepare an input flist for *reauthorization*, use the PCM_OP_GPRS_AAA_POL_REAUTHORIZE_PREP_INPUT helper opcode. See "Preparing GPRS-Specific Flists for Reauthorization".
- To prepare an input flist for *updating* a GPRS session, use the PCM_OP_GPRS_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT helper opcode. See "Preparing GPRS-Specific Flists for Updating Sessions".
- To prepare an input flist for *ending* a GPRS session, use the PCM_OP_GPRS_AAA_POL_STOP_ACCOUNTING_PREP_INPUT helper opcode. See "Preparing GPRS-Specific Flists for Ending Sessions".
- To build a search template for finding **/active_session/telco/gprs** or **/event/session/telco/gprs** objects, use the PCM_OP_GPRS_AAA_POL_SEARCH_SESSION helper opcode. See "Building Search Templates for GPRS Session Objects".
- To build a search template for finding **/active_session/telco/gprs** objects, use the PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH helper opcode. See "Building Search Templates for GPRS Active Session Objects".

Preparing GPRS-Specific Flists for Authorization

Use the PCM_OP_GPRS_AAA_POL_AUTHORIZE_PREP_INPUT helper opcode to aggregate GPRS data and then prepare an flist for authorizing a prepaid GPRS session. By default, this opcode is called by PCM_OP_TCF_AAA_AUTHORIZE at the PREP_INPUT processing stage.

This opcode aggregates GPRS data by amount, duration, volume, or occurrence, depending on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies that the events were *prerated*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Prerated Events:

When aggregating prerated events, the helper opcode prepares the PIN_FLD_BALANCES array in the PIN_FLD_RATING_INFO substruct, indexed by the currency type.

For Duration-Based Aggregation:

When aggregating the duration, the helper opcode performs the following:

1. Assigns a starting timestamp to the PIN_FLD_START_T field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct. The opcode uses the starting timestamp from the input flist or, if one is not passed in, from "pin_virtual_time" (see *BRM Developer's Guide*).

   Note: This is a temporary starting timestamp only and is later replaced with the actual starting timestamp by the reauthorization or stop accounting opcode.

2. Assigns an ending timestamp to the PIN_FLD_END_T field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct:
   - If PIN_FLD_END_T is supplied in the input flist, the opcode assigns the ending timestamp directly to the PIN_FLD_END_T field.
   - If PIN_FLD_QUANTITY is supplied in the input flist, the opcode calculates the ending timestamp by adding the quantity passed in the input flist to the starting timestamp.
   - If an ending timestamp or quantity is not passed in, the opcode retrieves the default authorization quantity from the **/config/reserve/gprs** object. The opcode calculates the ending timestamp by adding the default authorization quantity to the starting timestamp.

For Volume-Based Aggregation:

When aggregating the volume, the helper opcode adds the bytes uploaded and the bytes downloaded and assigns the value to the PIN_FLD_BYTES_UPLINK and PIN_FLD_BYTES_DOWNLINK fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT.PIN_FLD_TELCO_INFO substruct.

Preparing GPRS-Specific Flists for Reauthorization

Use the PCM_OP_GPRS_AAA_POL_REAUTHORIZE_PREP_INPUT helper opcode to aggregate GPRS data and then prepare an flist for reauthorizing a prepaid GPRS session. By default, this opcode is called by PCM_OP_TCF_AAA_REAUTHORIZE at the PREP_INPUT processing stage.

This opcode aggregates GPRS data by amount, duration, volume, or occurrence, depending on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies that the events were *prerated*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Prerated Events:

When aggregating prerated events, the helper opcode assigns a reauthorization amount to the PIN_FLD_AMOUNT field in the PIN_FLD_RATING_INFO substruct. The method the opcode uses to calculate the reauthorization amount depends on the value passed in the PIN_FLD_AGGREGATE_MODE field:

- If PIN_FLD_AGGREGATE_MODE is set to **4**, the opcode uses the amount passed in the input flist.
- If PIN_FLD_AGGREGATE_MODE is set to **8**, the opcode calculates the reauthorization amount based on the PIN_FLD_RATING_MODE field:
  - If PIN_FLD_RATING_MODE is **0**, the opcode adds the amount passed in the input flist to the amount in the **/active_session** object.
  - If PIN_FLD_RATING_MODE is **1**, the opcode adds the amount passed in the input flist to the amount in the **/reservation/active** object.

For Duration-Based Requests:

When aggregating the duration, the helper opcode assigns a starting timestamp and ending timestamp to the PIN_FLD_START_T and PIN_FLD_END_T fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct.

The opcode assigns a starting timestamp based on the following:

- If PIN_FLD_START_T is passed in the input flist and the timestamp is earlier than the timestamp in the **/active_session** object, the opcode assigns the starting timestamp from the input flist.
- If PIN_FLD_START_T is passed in the input flist and the **/active_session** object does not already exist, the opcode assigns the starting timestamp from the input flist.
- If PIN_FLD_START_T is not passed in and the **/active_session** object does not already exist, the opcode assigns the starting timestamp from "pin_virtual_time" (see *BRM Developer's Guide*).

The opcode assigns an ending timestamp based on the following:

- If PIN_FLD_END_T is passed in the input flist, the opcode assigns the ending timestamp from the input flist.

- If PIN_FLD_QUANTITY is passed in the input flist, the opcode calculates the ending timestamp based on the values passed in the PIN_FLD_AGGREGATE_MODE field:
  - If PIN_FLD_AGGREGATE_MODE is **4**, the opcode calculates the ending timestamp by adding the starting timestamp and the quantity.
  - If PIN_FLD_AGGREGATE_MODE is **8**, the opcode populates the value based on the PIN_FLD_RATING_MODE field:
    * If PIN_FLD_RATING_MODE is **0**, the opcode calculates the ending timestamp by adding together the ending timestamp from the **/active_session** object and the quantity from the input flist.
    * If PIN_FLD_RATING_MODE is **1**, the opcode calculates the ending timestamp by adding together the starting timestamp, the reserved quantity from **/reservation/active**, and the quantity from the input flist.
- If an ending timestamp or quantity is not passed in, the opcode retrieves the default reauthorization quantity from the **/config/reserve/gprs** object. The opcode calculates the ending timestamp by adding together the default reauthorization quantity and the starting timestamp.

For Volume-Based Requests:

When aggregating the volume, the helper opcode assigns the reauthorization volume to the PIN_FLD_BYTES_UPLINK and PIN_FLD_BYTES_DOWNLINK fields in the PIN_FLD_RATING_INFO substruct. The method the opcode uses to calculate the reauthorization volume depends on the value passed in the PIN_FLD_AGGREGATE_MODE field:

- If PIN_FLD_AGGREGATE_MODE is set to **4**, the opcode assigns the volume passed in the input flist.
- If PIN_FLD_AGGREGATE_MODE is set to **8**, the opcode calculates the reauthorization volume based on the PIN_FLD_RATING_MODE field:
  - If PIN_FLD_RATING_MODE is set to **0**, the opcode adds the bytes uploaded or downloaded from the input flist to the value in the **/active_session** object.
  - If PIN_FLD_RATING_MODE is set to **1**, the opcode adds the bytes uploaded or downloaded from the input flist to the value in the **/reservation/active** object.

Preparing GPRS-Specific Flists for Updating Sessions

Use the PCM_OP_GPRS_AAA_POL_UPDATE_ACCOUNTING_PREP_INPUT helper opcode to aggregate GPRS data and then prepare an flist for updating an existing prepaid GPRS session.

This opcode aggregates GPRS data by amount, duration, volume, or occurrence, depending on the value passed in the PIN_FLD_REQ_MODE flist field:

- **1** specifies that the events were *prerated*.
- **2** specifies to rate the *duration*. This is the default.
- **4** specifies to rate the *volume*.
- **6** specifies to rate the *duration and volume*.
- **8** specifies to rate the *occurrence*. This applies to activity events only.

For Prerated Events:

When aggregating prerated events, the helper opcode prepares the PIN_FLD_BALANCES array in the PIN_FLD_RATING_INFO substruct, indexed by the currency type.

- If PIN_FLD_AGGREGATE_MODE is **4**, the opcode assigns the amount from the input flist.
- If PIN_FLD_AGGREGATE_MODE is **8**, the opcode calculates the amount by adding together the amount passed in the input flist and the amount from the **/active_session** object.

For Duration-Based Requests:

When aggregating the duration, the helper opcode assigns a starting timestamp and ending timestamp to the PIN_FLD_START_T and PIN_FLD_END_T fields in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT substruct.

- For the starting timestamp, the helper opcode assigns the timestamp from either the **/active_session** object or the input flist, whichever is earlier. If a starting timestamp is not present in the input flist or the **/active_session** object, the helper opcode does not modify the existing starting timestamp.
- For the ending timestamp, the helper opcode assigns the timestamp from either the **/active_session** object or the input flist, whichever is later. If an ending timestamp is not present in the input flist or the **/active_session** object, the helper opcode does not modify the existing ending timestamp.

For Volume-Based Requests:

When aggregating the volume, the helper opcode assigns the volume uploaded or downloaded by the customer to the PIN_FLD_BYTES_UPLINK or PIN_FLD_BYTES_DOWNLINK field in the PIN_FLD_RATING_INFO.PIN_FLD_EVENT.PIN_FLD_TELCO_INFO substruct. The method the helper opcode uses to calculate the volume depends on the value passed in the PIN_FLD_AGGREGATE_MODE flist field:

- If PIN_FLD_AGGREGATE_MODE is **4**, the opcode uses the bytes passed in the input flist.
- If PIN_FLD_AGGREGATE_MODE is **8**, the opcode calculates the volume by adding together the bytes passed in the input flist and the bytes in the **/active_session** object.

Preparing GPRS-Specific Flists for Ending Sessions

Use the PCM_OP_GPRS_AAA_POL_STOP_ACCOUNTING_PREP_INPUT helper opcode to aggregate GPRS data and then prepare an flist for ending a prepaid GPRS session.

This opcode aggregates GPRS data based on the value passed in the PIN_FLD_SUBSESSION_MODE flist field:

- **1** specifies *aggregate mode*.
- **2** specifies *rate mode*.
- **3** specifies deferred rate mode.

For more information about the rating modes, see "Specifying How to Rate Subsessions".

For Aggregate Mode:

When set to aggregate mode, the helper opcode reads the PIN_FLD_STOP_INDICATOR field to determine whether the session is still in progress (**0**) or has ended

(**1**). When the field is set to **0**, the helper opcode does nothing. When the field is set to **1**, the helper opcode performs the following:

1. Aggregates the amount or quantity passed in the input flist with the amount or quantity in the **/active_session** object.
2. Aggregates the volume information from all of the subsession objects and records it in the master session object.
3. Determines the session's starting timestamp and ending timestamp by choosing the earliest and latest timestamps from all of the subsession objects and then records them in the master session object.
4. Sets the status of all subsession objects to **Closed And Unrated**, which indicates that the objects should be closed and the event should not be recorded.

If the object is a master session object, the information is passed in the top level of the flist; if it is a subsession object, the information is passed in the PIN_FLD_SESSION_INFO array.

For Rate Mode:

When set to rate mode, the helper opcode reads the PIN_FLD_STOP_INDICATOR field to determine whether the session is still in progress (**0**) or has ended (**1**). When the field is set to **0**, the helper opcode does nothing. When the field is set to **1**, the helper opcode performs the following:

1. Aggregates the amount or quantity passed in the input flist with the amount or quantity in the **/active_session** object.
2. Passes the **/active_session** object with the status set to **Closed**.

If the object is a master session object, the information is passed in the top level of the flist; if it is a subsession object, the information is passed in the PIN_FLD_SESSION_INFO array.

For Deferred Rate Mode:

When set to deferred rate mode, the helper opcode reads the PIN_FLD_STOP_INDICATOR field to determine whether the session is still in progress (**0**) or has ended (**1**). When the field is set to **0**, the helper opcode does nothing. When the field is set to **1**, the helper opcode performs the following:

1. Aggregates the amount or quantity passed in the input flist with the amount or quantity in the **/active_session** object.
2. Passes all of the master and subsession objects with the status set to **Closed**, which indicates that all of the objects should be rated and recorded as events.

If the object is a master session object, the information is passed in the top level of the flist; if it is a subsession object, the information is passed in the PIN_FLD_SESSION_INFO array.

Building Search Templates for GPRS Session Objects

Use the PCM_OP_GPRS_AAA_POL_SEARCH_SESSION helper opcode to build a search template for finding **/active_session/telco/gprs** objects or **/event/session/telco/gprs** objects. These search templates are used by the Services Framework AAA opcodes to look for duplicate session objects.

By default, this helper opcode sets the search criteria to the following, but you can customize it to use other criteria:

- Authorization ID

- APN name
- GGSN address
- SGSN address
- Starting timestamp
- For **/active_session/telco/gprs** objects, the authorization ID
- For **/event/session/telco/gprs** objects, the network session ID, MSISDN, and start time

Building Search Templates for GPRS Active Session Objects

Use the PCM_OP_GPRS_AAA_POL_ACC_ON_OFF_SEARCH helper opcode to build a search template that can be used to find **/active_session/telco/gprs** objects when the external network shuts down or restarts.

By default, this helper opcode uses the following search criteria, but you can customize it to use custom search criteria:

- Status of the **/active_session** objects
- GGSN address
- SGSN address
- (Optional) Starting timestamp

26

Installing GPRS Manager 3.0

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) GPRS Manager 3.0 software. The GPRS Manager 3.0 package includes both GPRS Manager 3.0 and a subset of Services Framework Manager, which is installed as a dependent component during installation.

Important: GPRS Manager 3.0 is an optional feature that requires a separate license.

Before you read this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

Installing GPRS Manager 3.0

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 GPRS Manager 3.0:

1. Download the software.

   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 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.

> **Caution:** 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 the temp_dir directory and enter this command:

   7.5.0_GPRS_Mgr_30_*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 GPRS Manager 3.0 is **opt/portal/7.5**.

   > **Note:** The installation program does not prompt you for the installation directory if BRM or GSM Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the GPRS Manager 3.0 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.

7. If the event tables of your BRM database are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home***/apps/partition_utils** directory:

   ```
   perl partition_utils.pl -o update
   ```

   For more information, see "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Your GPRS Manager 3.0 installation is now complete.

Configuring Event Notification for GPRS Manager

Important: This is a mandatory configuration task.

Before you can use GPRS Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them. See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Ensure that the merged file includes the entire event notification list in the *BRM_Home*/**sys/data/config/pin_notify_telco** file.
3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list. See "Editing the Event Notification List" in *BRM Developer's Guide*.
4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger. See "Triggering Custom Operations" in *BRM Developer's Guide*.
5. Load your final event notification list into the BRM database. See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Uninstalling GPRS Manager 3.0

To uninstall GPRS Manager 3.0, run the *BRM_Home*/**uninstaller/GPRS_Mgr_30/uninstaller.bin**.

27

About Managing and Provisioning GPRS Services

This chapter explains how to manage and provision GPRS services by using Oracle Communications Billing and Revenue Management (BRM) GPRS Manager 3.0.

Before you read this document, you should be familiar with the following:

- Global System for Mobile Communication (GSM) and General Packet Radio Service (GPRS) network terms and concepts.
- European Standards Technical Institute (ETSI) GSM Technical Specifications for GPRS, document ETSI TS 101 393 V7.6.0 (2000-11).
- BRM concepts and architecture. See *BRM Concepts*.

About GPRS Manager 3.0

You use GPRS Manager 3.0 to provision and manage your GPRS services. GPRS Manager 3.0 is an API that consists of opcodes, storable classes, and utilities that allow you to easily set up GPRS provisioning.

GPRS Manager 3.0 runs on top of the Services Framework Manager, which is an abstract framework for provisioning and managing any prepaid service type. Because Services Framework Manager is an abstract framework, it cannot perform GPRS-specific operations by itself. It relies upon the GPRS opcodes and objects to perform these tasks.

Note: A subset of Services Framework Manager is shipped with GPRS Manager 3.0.

About Provisioning GPRS Services

Provisioning occurs whenever account management actions, such as activating, changing, and inactivating GPRS services, require changes on the carrier network. For example, during GPRS service activation and inactivation, email addresses can be provisioned or unprovisioned.

When customers purchase or update their GPRS services, events occur that trigger wireless service provisioning:

1. BRM generates a service order and sends it to the Provisioning Data Manager (DM). The service order contains the information required for service provisioning.

2. The Provisioning Data Manager processes the request and converts the service order information into flist XML format. The service order is then sent to a third-party provisioning system.

For more detailed information, see "About Provisioning Telco and Non-Telco Services".

About Associating APNs and QoS with GPRS Services

In BRM, an APN is stored as a device (**/device/apn**). Because the APN is a network-level device and because a single APN can be used by multiple customers, you cannot map APN devices to GPRS services. Instead, the GPRS service references an APN and its appropriate quality of service (QoS) value based on the product purchased by the customer.

BRM associates APN and QoS pairs with a GPRS service as follows:

1. A customer purchases, cancels, or modifies a deal or product that includes a GPRS service and provisioning tag.
2. The BRM system generates one of the following business events:
   - **/event/billing/deal/purchase**
   - **/event/billing/production/action/modify**
   - **/event/billing/deal/cancel**
   - **/event/billing/product/action/cancel**
   - **/event/billing/product/action/purchase**
3. The BRM event notification system detects the event and calls the opcode specified in the **/config/notify** object. The default configuration specifies to call PCM_OP_TCF_SVC_LISTENER.
4. PCM_OP_TCF_SVC_LISTENER checks the event's start and end date to determine whether the action is deferred for a future date.
   - If the event *is not* deferred, the opcode calls the PCM_OP_TCF_APPLY_PARAMETER opcode to update the GPRS service and ERA objects impacted by the product provisioning update.
   - If the event *is* deferred, the opcode creates a **/schedule** object for executing the PCM_OP_TCF_APPLY_PARAMETER opcode at the scheduled time.
5. PCM_OP_TCF_APPLY_PARAMETER retrieves the service extension and service ERA information from the **/config/telco/gprs** object and passes the information to PCM_OP_TCF_POL_APPLY_PARAMETER.
6. When the service type is **/service/telco/gprs**, PCM_OP_TCF_POL_APPLY_PARAMETER calls PCM_OP_GPRS_APPLY_PARAMETER.
7. PCM_OP_GPRS_APPLY_PARAMETER writes the APN and QoS pairs to the flist's PIN_FLD_APN_ARRAY array.
8. PCM_OP_GPRS_APPLY_PARAMETER calls PCM_OP_GPRS_POL_APPLY_PARAMETER to update information about any custom **/service/telco/gprs** fields.
9. PCM_OP_TCF_APPLY_PARAMETER updates the service extensions and ERAs in the **/service/telco/gprs** object.

To customize how BRM associates APNs and QoS values with GPRS services, see "Associating APNs and QoS with GPRS Services".

Setting Up Provisioning for GPRS Services

To set up provisioning for GPRS services, perform the following tasks:

1. Creating Provisioning Tags for GPRS Services
2. Specifying the Provisioning Configuration for GPRS Services
3. Specifying the Available States for Each GPRS Service Order
4. Specifying the Event Types Available for GPRS Services
5. Creating RUMs for GPRS Services

Creating Provisioning Tags for GPRS Services

To implement service extensions, such as bearer services, APNs, and QoS values, and extended rating attributes (ERAs) for GPRS services, you define provisioning tags. You then use Pricing Center to include provisioning tags in products. A tag becomes available to an individual account and service when a product containing the tag is purchased. This is also known as product-level provisioning.

For more information, see "About Provisioning Tags for Telco Services".

To define provisioning tags for GPRS services, you modify the **pin_telco_tags_gprs** file. You then load the file into the BRM database's **/config/telco/gprs** and **/config/account_era** objects by using the **load_pin_telco_tags** utility.

> **Note:** You can also use the Provisioning Tags application in Pricing Center instead of the **pin_telco_tags** file to define provisioning tags for prepaid services. But you cannot create custom ERAs using Provisioning Tags. For information, see Provisioning Tags Help.

To specify provisioning tags for GPRS services, perform the following:

1. Open the *BRM_Home*/**sys/data/config/pin_telco_tags_gprs** file in a text editor.
2. If necessary, edit the **pin_telco_tags_gprs** file. The sample file includes these entries:

```
provisioning_tag    "/config/telco/gprs "    "Data Premium"    "Data Service"
"y"
service_extn       "BEARER_SERVICE"    "BS 70"
service_extn       "APN"               "apn.portal.com"
service_extn       "QOS"               "Platinum"
service_era        "FRIENDS_FAMILY"    12 13 "n"
service_era        "HOME_CELL"         16 17 "y"
service_era        "HOME_REGION"       14 15 "y"
service_era        "SERVICELEVEL"      10 11 "n"
service_era        "RATEPLAN"          18 19 "n"
```

> **Note:** List the APN and QoS entries in order, so that the PIN_FLD_APN_ARRAY is populated by taking the APN name along with its associated QoS. In addition, the QoS entry is optional for each APN.

   For more information on how to edit the file, see "Configuring Provisioning Tags in the pin_telco_tags File".

3. Save the **pin_telco_tags_gprs** file.

4. Use the following command to run the **load_pin_telco_tags** utility:

   ```
   load_pin_telco_tags pin_telco_tags_gprs
   ```

   For the complete command syntax, see "load_pin_telco_tags".

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

6. Restart Pricing Center.

To verify that the account ERAs were loaded, you can display the **/config** objects by using the Object Browser or use the **robj** command with the **testnap** utility. (See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.)

Specifying the Provisioning Configuration for GPRS Services

The provisioning configuration data includes information about which fields need to be included to create a service order for GPRS services. You specify the configuration in the GPRS provisioning configuration file (**pin_telco_provisioning_gprs**) and then load the file in the BRM database's **/config/telco/provisioning** object by using the "load_pin_telco_provisioning" utility.

To specify the provisioning configuration information for GPRS services, perform the following:

1. Open the *BRM_Home*/**sys/data/config/pin_telco_provisioning_gprs** file in a text editor.

2. If necessary, edit the **pin_telco_provisioning_gprs** file. The default file includes these entries:

```
Service provisioning info:
/service/telco/gprs, A,
PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE
PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME

Service provisioning info:
/service/telco/gprs, C,
PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE
PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME

Service provisioning info:
/service/telco/gprs, D,
PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE
PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME

Service provisioning info:
/service/telco/gprs, R,
PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE
PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME

Service provisioning info:
/service/telco/gprs, S,
PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE
PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME
```

```
Service provisioning info:
/service/telco/gprs, I,
PIN_FLD_GPRS_INFO.PIN_FLD_BEARER_SERVICE
PIN_FLD_APN_ARRAY[*].PIN_FLD_APN,
PIN_FLD_APN_ARRAY[*].PIN_FLD_QOS_PROFILE_NAME
```

For more information on how to edit the file, see "load_pin_telco_provisioning".

3. Save the **pin_telco_provisioning_gprs** file.
4. Use the following command to run the **load_config_provisioning_tags** utility:

   ```
   load_pin_telco_provisioning pin_telco_provisioning_gprs
   ```

   For the complete command syntax, see "load_pin_telco_provisioning".
5. Restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the account ERAs were loaded, you can display the **/config** objects by using the Object Browser or use the **robj** command with the **testnap** utility. (See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.)

Specifying the Available States for Each GPRS Service Order

You specify the available states for GPRS service orders by editing the **pin_telco_service_order_state_gprs** file. You then load the file into the BRM database's **/config/telco/service_order_state/gprs** object by using the **load_pin_telco_service_order_state** utility.

See "About Service Order Status" for more information about service order states.

To specify the available states, perform the following:

1. Open the *BRM_Home*/**sys/data/config/pin_telco_service_order_state_gprs** file in a text editor.
2. If necessary, edit the entries to support your business needs. The default entries are shown below:

```
1: 1: 0: 0
      2: 0:0
      3: 0:0
      4: 0:0
      5: 0:0
# READY -> PROCESSING
2: 2: 0: 0
      1: 0: 0
      2: 0: 0
      3: 0: 0
      4: 0: 0
      5: 0: 0
# PROCESSING -> FAILED or COMPLETED
3: 2: 0: 0
      1: 0: 0
      2: 0: 0
      3: 0: 0
      4: 0: 0
      5: 0: 0
# Completed Provisioning -> Completed Provisioning (terminating state)
4: 3: 0: 0
      1: 0: 0
      2: 0: 0
```

```
      3: 0: 0
      4: 0: 0
      5: 0: 0
# Failed Provisioning -> Failed Provisioning (terminating state)
5: 3: 0: 0
      1: 0: 0
      2: 0: 0
      3: 0: 0
      4: 0: 0
      5: 0: 0
```

For more information about editing the input file, see "Specifying the Available States for Each Service Order".

3. Save and close the **pin_telco_service_order_state_gprs** file.
4. Use the following command to run the **load_pin_telco_service_order_state** utility:

   ```
   load_pin_telco_service_order_state pin_telco_service_order_state_gprs
   ```

   For the complete command syntax, see "load_pin_telco_service_order_state".
5. Restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the account ERAs were loaded, you can display the **/config** objects by using the Object Browser or use the **robj** command with the **testnap** utility. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Specifying the Event Types Available for GPRS Services

Specify the event types available for GPRS services or accounts when creating products.

To map event types to GPRS services, you edit the **pin_event_map** file and then run the **load_event_map** utility to load the contents of the file into the **/config/event_map** object in the BRM database.

> **Caution:** The **load_event_map** utility overwrites the existing event map. If you are updating the event map, you cannot load new mappings only. You must load the entire event map each time you run the **load_event_map** utility.

To map event types to GPRS services, perform the following:

1. Open the *BRM_Home*/**sys/data/pricing/example/pin_event_map_telco_gprs** file in a text editor.
2. If necessary, edit the **pin_event_map_telco_gprs** file. The default file includes the following entries.

```
#================================================
# Syntax:
# <purchase_level> : <event_type> : <event_description> : <count>
#================================================
#================================================
# Telco GPRS events
#================================================
/service/telco/gprs: /event/session/telco/gprs/master: Telco GPRS master
session
```

```
: /event/session/telco/gprs/subsession: Telco GPRS subsession
#================================================
# DELAYED Telco GPRS events
#================================================
service/telco/gprs : /event/delayed/session/telco/gprs: Delayed Telco GPRS
session
```

For more information on how to edit the file, see "Mapping Event Types to Services".

3. Save the **pin_event_map** file.
4. Use the following command to run the **load_event_map** utility:

```
load_event_map pin_event_map_gprs
```

> **Note:** If you are not in the same directory as the **pin_event_map_gprs** file, include the complete path to the file.

For more information, see "load_event_map" in *BRM Setting Up Pricing and Rating*.

5. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*. If necessary, stop and restart Pricing Center.

To verify that the **pin_event_map** file was loaded, you can display the **/config/event_map** 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*.)

Creating RUMs for GPRS Services

You can define a list of ratable usage metrics (RUM) available for each event type. Each RUM consists of an event type to be rated, a name, how the event should be quantified, and the units used to quantify it. You can specify more than one RUM for a given event type.

To create RUMs, edit the **pin_rum** file and then run the **load_pin_rum** utility to load the contents of the file into the **/config/rum** object in the BRM database.

> **Caution:** The **load_pin_rum** utility overwrites existing RUMs. If you are updating RUMs, you cannot load new RUMs only. You must load complete sets of RUMs each time you run the **load_pin_rum** utility.

To create ratable usage metrics:

1. Open the *BRM_Home***/sys/data/pricing/example/pin_rum** file in a text editor.
2. Edit the **pin_rum** file. The default file includes the following entries for GPRS services:
3. The entries in the **pin_rum** configuration file for gprs will look as follows:

```
/event/session/telco    : Size     : (PIN_FLD_TELCO_INFO.PIN_FLD_BYTES_
UPLINK+PIN_FLD_TELCO_INFO.PIN_FLD_BYTES_DOWNLINK) : byte

/event/session/telco    : UpLink : PIN_FLD_TELCO_INFO.PIN_FLD_BYTES_UPLINK :
byte
```

```
/event/session/telco    : Downlink   : PIN_FLD_TELCO_INFO.PIN_FLD_BYTES_
DOWNLINK
```

4. Save and close the **pin_rum** file.
5. Use the following command to run the **load_pin_rum** utility:

   ```
   load_pin_rum pin_rum_file
   ```

   If you are not in the same directory as the **pin_rum** file, include the complete path to the file.

   For more information, see "load_pin_rum" in *BRM Setting Up Pricing and Rating*.
6. Stop and restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*. If necessary, stop and restart Pricing Center.

To verify that the **pin_event_map** file was loaded, you can display the **/config/event_map** 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: Fold events cannot use custom RUMs; therefore, do not assign custom RUMs to fold events in any rate plans. Products configured with custom fold RUMs are rated incorrectly.

Associating APNs and QoS with GPRS Services

To associate APNs and QoS values with GPRS services, you use the following opcodes:

- To map service types to specific APPLY_PARAMETER opcodes, use PCM_OP_TCF_POL_APPLY_PARAMETER. See "Mapping Service Types to Service-Specific Opcodes".
- To add APN and QoS pairs for GPRS services, use PCM_OP_GPRS_APPLY_PARAMETER. See "Associating APN and QoS Pairs with GPRS Services".
- To update custom fields in GRPS services, use PCM_OP_GPRS_POL_APPLY_PARAMETER. See "Updating Custom GPRS Service Fields".

Mapping Service Types to Service-Specific Opcodes

Use the PCM_OP_TCF_POL_APPLY_PARAMETER policy opcode to update information in the flist's PIN_FLD_SERVICES and PIN_FLD_PRODUCTS array and then pass the information to the appropriate service-specific opcode.

Table 27–1 shows the opcode called for each supported service type:

Table 27–1 Supported Service Type Opcodes

Service Type	Opcode Called
/service/telco/gprs	PCM_OP_GPRS_APPLY_PARAMETER
/service/telco/gsm	PCM_OP_GSM_APPLY_PARAMETER

Associating APN and QoS Pairs with GPRS Services

Use the PCM_OP_GPRS_APPLY_PARAMETER opcode to associate APN and QoS pairs with a **/service/telco/gprs** service. This opcode is called by the PCM_OP_TCF_POL_APPLY_PARAMETER opcode when processing **/service/telco/gprs** services.

PCM_OP_GPRS_APPLY_PARAMETER reads the bearer service, APN name, and QoS information from the input flist's PIN_FLD_SERVICE_EXTENSIONS array and performs the following:

- If the Bearer service is passed in the input flist, the opcode adds the value to the output flist's PIN_FLD_BEARER_SERVICE field of the PIN_FLD_GPRS_INFO substruct.
- If the APN name and QoS are passed in the input flist, the opcode adds the values to the output flist's PIN_FLD_APN array in the PIN_FLD_INHERITED_INFO substruct.

The opcode then calls the PCM_OP_GPRS_POL_APPLY_PARAMETER policy opcode to perform any customizations. See "Updating Custom GPRS Service Fields".

Updating Custom GPRS Service Fields

Use PCM_OP_GPRS_POL_APPLY_PARAMETER to update custom fields in the **/service/telco/gprs** object. This policy opcode takes as input the configuration object flist, the service flist, and the inherited information flist from the calling PCM_OP_GPRS_APPLY_PARAMETER opcode.

By default, this policy opcode returns the information passed in the input flist. This policy opcode can be customized to update the service flist by adding values to customized fields.

28

Installing GPRS AAA Manager

This chapter explains how to install the Oracle Communications Billing and Revenue Management (BRM) GPRS AAA Manager software. The GPRS AAA Manager package includes both GPRS AAA Manager and a subset of Services Framework AAA Manager, which is installed as a dependent component during installation.

Important: GPRS AAA Manager is an optional feature that requires a separate license.

Before you read this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

System Requirements

GPRS AAA Manager is available for the HP-UX IA64, Solaris, AIX, and Linux operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Software Requirements

Before installing GPRS AAA Manager, install the following software:

- Third-Party software, which includes the PERL 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 10*g* or Oracle 11*g*.
- Resource Reservation Manager.

Note: If you offer concurrent use of multiple GPRS services or multiple GPRS sessions for a single customer account, you should also install Resource Reservation Manager before running GPRS AAA Manager. GPRS AAA Manager does not require Resource Reservation Manager to support concurrent sessions; however, without Resource Reservation Manager, you may encounter revenue leakage. For information about Resource Reservation Manager, see "Reserving Resources for Concurrent Network Sessions" in *BRM Configuring and Collecting Payments*.

Installing GPRS AAA 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 GPRS AAA 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 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.

   > **Caution:** 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 the temp_dir directory and enter this command:

   ```
   7.5.0_GPRS_AAA_Mgr_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 GPRS AAA Manager is **opt/portal/7.5**.

> **Note:** The installation program does not prompt you for the installation directory if BRM or GPRS AAA Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the GPRS AAA 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.

7. If your event tables are partitioned, run the **partition_utils** utility with the **-o update** parameter from the *BRM_Home*/**apps/partition_utils** directory:

   ```
   perl partition_utils.pl -o update
   ```

   For more information, see "Updating Partitions" and "partition_utils" in *BRM System Administrator's Guide*.

Your GPRS AAA Manager installation is now complete.

Uninstalling GPRS AAA Manager

To uninstall GPRS AAA Manager, run the *BRM_Home*/**uninstaller/GPRS_AAA_Mgr/uninstaller.bin**.

Part VI

Managing IMT and PDC Services

Part VI describes how to manage IMT and PDC services in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

29

About Using IMT Manager

This chapter provides an overview of how to use your Oracle Communications Billing and Revenue Management (BRM) system and IMT Manager to rate IMT (International Mobile Telecommunications) and PDC (Personal Digital Cellular) services.

Before reading this document, you should be familiar with BRM concepts and architecture. See *BRM Concepts*.

Important: IMT Manager is an optional component, not part of base BRM.

About Using IMT Manager

You use IMT Manager to rate and manage IMT and PDC service usage. IMT Manager provides the following features:

- Definitions of basic IMT and PDC services and events.
- Support for configuring promotions and extended rating attributes (ERAs).

To rate IMT and PDC services using BRM, you perform the following tasks:

- Install the following optional components:
  - Pipeline Rating Engine
  - IMT Manager
- Extend the basic IMT and PDC service classes and implement your policies by customizing the policy opcodes. For example, to support voice service for IMT, you need to extend the basic IMT service.

  See "Adding a New IMT or PDC Service".
- Prepare the configuration files for your service and load the files into the BRM database.
- Test your installation and configuration.

Note: IMT Manager does *not* support the following features:

- Customer registration and management using Customer Center.
- Service provisioning.
- Device management.

IMT Manager uses the **Alias** field in the **/service** object to associate the service with the correct device for rating usage.

How Usage Events Are Rated

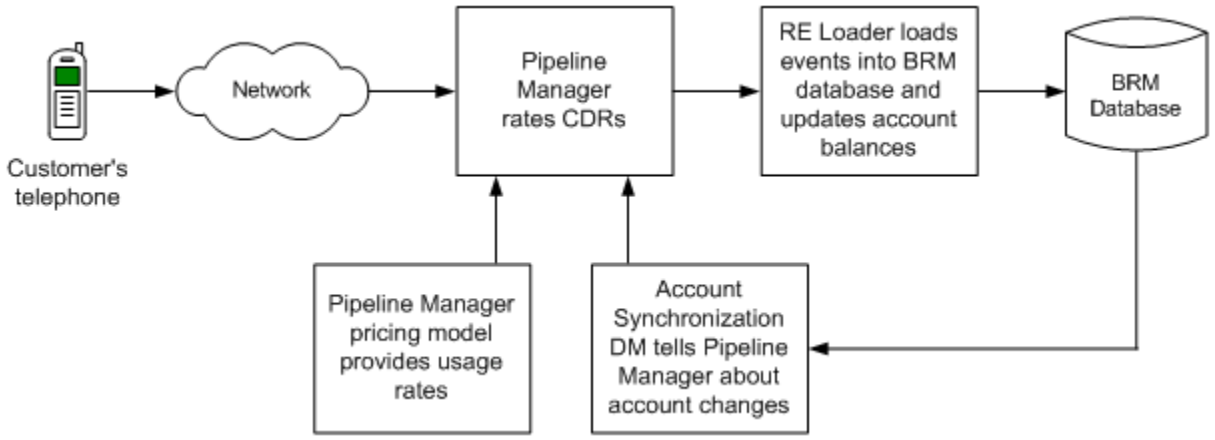
IMT and PDC networks send usage events in call detail records (CDRs) to the BRM system. CDRs for postpaid IMT and PDC services are sent in files to Pipeline Manager for rating:

1. The pipeline reads each CDR file and rates the CDR by using the following elements:
   - Pipeline rating rate plans
   - Customer data obtained from the BRM database by using the Account Synchronization Data Manager.

   Note: By default, pipeline rating identifies customer accounts by using the telephone number for the service being rated. You can customize pipeline rating to use any type of unique ID to identify accounts.
2. Pipeline Manager creates an output file that includes a pre-rated event for the call.
3. Rated Event (RE) Loader loads the event into the BRM database and updates the customer's account balance.

Figure 29–1 shows how a call is rated by the pipeline:

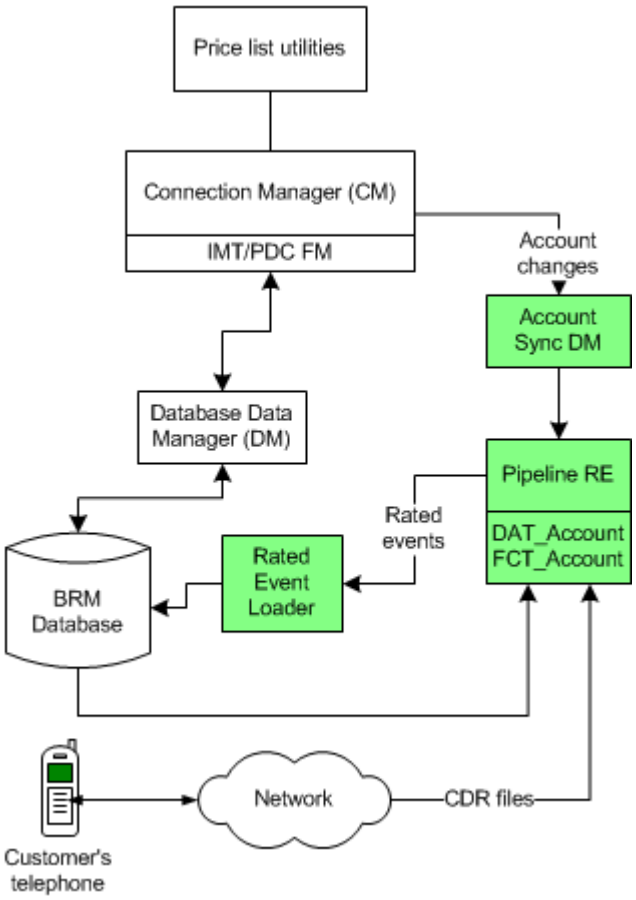
Figure 29–1 Usage Events Rating Process



Overview of a BRM Wireless System

Before you install and configure BRM to rate IMT and PDC services, you must understand how the wireless components in a BRM system work together. Figure 29–2 shows an overview of the BRM components in a system that supports IMT and PDC services:

Figure 29–2 IMT and PDC BRM Components



Setting Up the Price List

To set up your price list, you:

- Use Pricing Center to set up your IMT and PDC rate plans.
- Use the sample GSM service price lists as examples when creating your IMT and PDC price lists.

See "Setting Up GSM Wireless Pricing".

Setting Up Rating

You use the following components to rate calls:

- Use Pipeline Manager to rate events, such as calls or data transfers. Pipeline Manager uses the **DAT_AccountBatch** and **FCT_Account** modules to retrieve data from the BRM database and apply it to events when rating.
- Use the Account Synchronization Data Manager (DM) to send updated account information to the pipeline for call rating. See "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*.
- Use Rated Event (RE) Loader to import prerated events into BRM. See "Loading Prerated Events" in *BRM Configuring Pipeline Rating and Discounting*.

For more information, see "About Integrating Wireless Services".

30

Installing and Configuring IMT Manager

This chapter provides an overview of how components relate to one another in an Oracle Communications Billing and Revenue Management (BRM) wireless system and describes the tasks you need to perform to install and configure IMT Manager to work with BRM components.

For general information about IMT Manager, see "About Using IMT Manager".

For basic information about integrating wireless services with BRM, see "About Integrating Wireless Services".

Overview of IMT Manager Installation and Configuration Tasks

To install and use IMT Manager:

1. **Install and configure BRM**. This includes setting up standard BRM business policies, such as how to run billing and manage customers.
2. Set up your G/L IDs, rate plans, and resources before configuring pipeline rating. See "About Integrating Wireless Services".
3. **Install and configure IMT Manag**er. See "Installing IMT Manager" and "Configuring IMT Manager".
4. **Install Pipeline Manager.** See "Overview of Pipeline Manager Installation".
5. Extend the IMT and PDC service classes to offer IMT and PDC products and services to your users. See "Adding a new IMT or PDC service".

Important:

- Do not install IMT Manager and GSM Manager together in the same BRM system. There will be conflicts because they create the same set of services for two different telecommunications standards.
- Make sure you install, configure, and test each component before installing the next component. Some components need to be configured before others. See "Component Configuration Dependencies".
- Before installing and configuring any optional components, such as Service Manager, run enough tests to ensure that your BRM core server components are installed and configured correctly.

Overview of Pipeline Manager Installation

The Pipeline Manager installation process includes the following:

1. Installing and configuring Account Synchronization Data Manager (DM). The Account Synchronization DM enables the pipeline to get data from the BRM database.
2. Running the **object_auditing** script to turn on auditing for BRM objects that pipeline rating needs information about.

   See "object_auditing" and "About Sending Account Data to Pipeline Manager" in *BRM Installation Guide*.

   Important: You cannot complete all the tasks required for sending account data to pipeline rating until you install Pipeline Manager. For example, when you configure the Account Synchronization Data Manager, you need to specify the location of the Listener map file.

3. Install and configure Pipeline Manager. Before you configure the rating pipeline, use the sample registry to test the system and make sure it has been installed correctly. See "Installing Pipeline Manager" in *BRM Installation Guide*.
4. Configure these pipeline modules:
   - DAT_Listener
   - DAT_AccountBatch
   - FCT_Account
   - (Multischema systems only) FCT_AccountRouter

   See *BRM Configuring Pipeline Rating and Discounting*.
5. Install and configure Rated Event Loader to load events into the BRM database. See "Installing Rated Event Loader" in *BRM Configuring Pipeline Rating and Discounting*.

Summary of Installable Components

Table 30–1 summarizes the components you install for a typical wireless integration:

Table 30–1 Installable Components for a Typical Wireless Integration

Installation Package	Description
BRM server software	Installs the standard BRM system software, including Connection Managers (CMs) and Data Managers (DMs).
Pipeline Manager	Installs the Pipeline Manager system software, modules, database, and utilities. The FCT_Account and DAT_ AccountBatch modules are installed with Pipeline Manager.
IMT Manager	Installs the following: ■ IMT Manager opcodes and storable classes. ■ Wireless Provisioning Data Manager. ■ Wireless provisioning opcodes.
Account Synchronization Data Manager	Installs Account Synchronization Data Manager (DM), and **object_auditing** script.
Rated Event Loader	Installs Rated Event Loader.

Component Configuration Dependencies

Some components cannot be configured without configuring another component first:

- Before you install any optional BRM components, you must install the BRM server software.
- Before you create BRM provisioning tags for extended rating attributes (ERAs), you must configure ERAs in the pipeline.
- Before you create your IMT and PDC price lists, you must perform the following tasks:
  - Install IMT Manager.
  - Create and load BRM provisioning tags. To use the sample BRM IMT and PDC price lists, you can load the sample **pin_telco_tags_pdc_imt** file without editing it. See "Loading Provisioning Tags".
- Before you configure rating in the pipeline, you must define the following items in BRM:
  - Rate plan names. See "Creating Pipeline Rate Plans and Price Models" in *BRM Setting Up Pricing and Rating*.
  - G/L IDs. See "Creating G/L IDs and G/L Segments" in *BRM Collecting General Ledger Data*.
  - Resources. See "About Resources" in *BRM Setting Up Pricing and Rating*.

Prerequisites for Running Components

- Before you can run the Account Synchronization DM, you need to install and configure the Pipeline Manager. See "Installing Pipeline Manager" in *BRM Installation Guide*.
- Before you run the **object_auditing** script, you must install IMT Manager.
- Before you run the pipeline with the DAT_AccountBatch and FCT_Account modules, you must run the **object_auditing** script to create the audit event tables that the DAT_AccountBatch and FCT_Account modules need. See "Turning on Object Auditing" in *BRM Installation Guide*.

Supported Operating Systems and Databases

IMT Manager runs on HP-UX platform. Rated Event (RE) Loader can load events only into an Oracle database.

For detailed information on supported versions, see "BRM Software Compatibility" in *BRM Installation Guide*.

Running Components on Different Machines

The pipeline and RE Loader should run on the same UNIX system.

> **Note:** Although you can install RE Loader on the BRM system or the database system, you get better performance if you install it on the pipeline system.

If the pipeline and RE Loader are on different systems, you need to map the pipeline output directories to a drive local to RE Loader.

For a test system, you use two machines:

- BRM
- Pipeline Manager and Rated Event Loader

For a production system, you run BRM components on multiple machines. Follow the guidelines described in "Putting Together Your BRM System" in *BRM Installation Guide*.

Hardware and Software Requirements for IMT Manager

You can install IMT Manager on HP-UX.

Before installing IMT Manager, you must install BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.

IMT Manager installation requires 50 MB of disk space.

You need the following information to install IMT Manager:

- The URL, login name, and password for the BRM Software Web site from your BRM sales representative.
- Get the following information about your existing BRM system from the existing *BRM_Home*/**setup/pin_setup.values** file on your BRM server.
  - The location of your *BRM_Home* directory and log directories.
  - The BRM database and port number.
  - The SQL*Net alias name for Oracle databases or data source name for SQL Server databases.
  - The user name and password for the database used by your BRM Server software.

Installing IMT Manager

1. Download the IMT Manager software from the BRM Web site to a temporary directory (*temp_dir*).

   Note: f you downloaded to a Windows workstation, use **FTP** to copy the **.tar.Z** file to a temporary directory on your UNIX server. You also need an application such as WinZip for extracting compressed files.

   Tip: If you are installing multiple components, use a different temporary directory for each component. Otherwise, files for separate components might be overwritten.

2. Log in to your system as **root**.
3. Stop all BRM processes. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
4. Go to *temp_dir* and extract the IMT Manager files.

```
uncompress 7.0_IMTMgr_hpux.tar.Z
tar xvf 7.0_IMTMgr_hpux.tar
```

5. Go to *temp_dir*, run the installation script **install.ksh**, and answer the prompts.
6. Run the **pin_setup** script.

   Your IMT Manager installation is now complete.
7. Verify that the IMT Manager Facilities Modules (FMs) were added to the Connection Manager (CM) configuration file. See "Enabling IMT Manager and Disabling Provisioning".
8. Configure your IMT Manager. See "Configuring IMT Manager".

Configuring IMT Manager

Important: Install IMT Manager before starting these configuration tasks. See "Installing IMT Manager"

To configure your IMT Manager to work properly and to disable provisioning, perform these tasks described in the sections that follow:

- Applying the Correct Partitioning Layout to Event Tables
- Enabling IMT Manager and Disabling Provisioning
- Loading IMT Manager Configuration Files

For information about configuration files, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Applying the Correct Partitioning Layout to Event Tables

Important: If your existing event tables are not partitioned, skip this task.

When you install IMT Manager, you create additional event tables. If your event tables are partitioned, to ensure that the new event tables have the same partitioning layout as your existing event tables, enter this command:

```
partition_utils.pl -n
```

For more information, see "Partitioning Database Tables" in *BRM System Administrator's Guide*.

Enabling IMT Manager and Disabling Provisioning

To enable IMT Manager, you disable provisioning and make sure that the CM configuration file includes entries for IMT and PDC FMs (Facilities Modules).

1. Open the Connection Manager (CM) **pin.conf** file in *BRM_Home*/**sys/cm**.
2. Verify that the following entries are included:

```
- cm fm_module BRM_Home/lib/fm_tcf.sl fm_tcf_config fm_tcf_init pin
- cm fm_module BRM_Home/lib/fm_tcf_pol.sl fm_tcf_pol_config - pin
- cm fm_module BRM_Home/lib/fm_imt_pol.sl fm_imt_pol_config - pin
- cm fm_module BRM_Home/lib/fm_pdc_pol.sl fm_imt_pol_config - pin
- cm fm_module BRM_Home/lib/fm_prov.sl fm_prov_config fm_prov_init pin
```

```
- cm fm_module BRM_Home/lib/fm_prov_pol.sl fm_prov_pol_config - pin
```

> **Note:** These lines are added automatically when you install IMT Manager. If the entries do not exist, add them. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

1. To disable provisioning,
   a. Add the following entry:

      ```
      -fm_tcf provisioning_enabled 0
      ```

   b. Be sure the **dm_pointer** configuration line is commented out:

      ```
      //- cm dm_pointer 0.0.10.2 ip BRM42 --   dm_prov_telco
      //- fm_prov_telco prov_db 0.0.10.2 / 0
      ```

2. Save and close the **pin.conf** file.
3. Restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Loading IMT Manager Configuration Files

After installing and configuring IMT Manager, use the appropriate utilities to load the configuration files containing the following data:

- Provisioning tags
- The event map
- IMT and PDC notification events

Loading Provisioning Tags

Load provisioning tags by running the **load_pin_telco_tags** utility:

```
load_pin_telco_tags -dv pin_telco_tags_pdc_imt
```

> **Note:** The **pin_telco_tags_pdc_imt** file has provisioning tags for both PDC and IMT services.

For more information, see "load_pin_telco_tags".

Loading the Event Map

Load the telco event mapping file, **pin_event_map_imt**, by running the **load_event_map** utility.

For more information, see **load_event_map** in *BRM Setting Up Pricing and Rating*.

Loading IMT or PDC Notification Events

> **Important:** This is a mandatory configuration task.

To enable communication with the Pipeline Manager, BRM uses event notification. Event notification is defined in a notification list stored in the BRM database. This list maps BRM opcodes to the BRM events associated with specified events. When one of these BRM events occurs, the corresponding opcode is executed, which triggers the BRM functionality.

To add the PDC or IMT event notification list to the BRM database, you use the **load_pin_notify** utility to load the **pin_notify_telco_pdc_imt** file located in the *BRM_Home*/**sys/data/config** directory.

Note: The **load_pin_notify** utility requires a configuration file. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

Caution: The **load_pin_notify** utility overwrites all existing notification data in the BRM database. You must load all notification data each time you run the **load_pin_notify** utility.

To load the PDC or IMT notification events:

1. Use the following command to run the **load_pin_notify** utility:

   ```
   load_pin_notify pin_notify_telco_pdc_imt
   ```

   For more information, see "load_pin_notify" in *BRM Managing Customers*.

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

31

Configuring IMT and PDC Services and Extended Rating Attributes

This chapter describes how to set up PDC and IMT services, supplementary services, and extended rating attributes (ERAs) in your Oracle Communications Billing and Revenue Management (BRM) system.

For background information about wireless services, see "About Integrating Wireless Services".

For general information about IMT Manager, see "About Using IMT Manager".

For information on adding new IMT and PDC services, see "Adding a new IMT or PDC service".

For information about ERAs and supplementary services, see "About Extended Rating Attributes" and "About Supplementary Services".

> **Note:** The variable *service_name* in the following sections refers to IMT or PDC.

About PDC and IMT BRM Services

You use BRM services to manage real-world services in BRM. BRM services are identified by classes and objects, for example, **/service/telco/imt and /service/telco/pdc** identify IMT and PDC services.

When you create products in Pricing Center, you use provisioning tags to assign different types of services to your products. For more information, see "About Configuring Services, Supplementary Services, and ERAs".

Your price list can include specific deals for one or more BRM PDC and IMT services. For example, you might have a plan that includes just the PDC or IMT telephony service (for example, **/service/telco/IMT/telephony**) and a plan that includes all BRM PDC or IMT services. You can create different usage and subscription rates for different services.

For more information, see "Setting Up the IMT and PDC Price Lists".

About Supplementary Services

IMT and PDC supplementary services are features such as call forwarding and call blocking. They are not implemented as BRM services. Instead, they are implemented by using product-level provisioning. For example, a price list might include a product associated with the telephony service, with a provisioning tag that implements the call

forwarding supplementary service. Supplementary services can be used only with a base service, for example, the base IMT or PDC service.

Value-added services, such as voice mail, are similar to supplementary services. The difference is that value-added services are not part of the wireless network standard.

> **Note:** Supplementary services and value-added services are handled in the same way by BRM. Therefore, this documentation uses the term supplementary services to include value-added services.

Supplementary services are included in products by using provisioning tags. You can create products that add supplementary services to existing accounts, for example, a product that adds call forwarding to an existing telephony service.

After BRM provisions a supplementary service, it is usually activated by the customer using their telephone keypad, for example, the customer can define the number keys to press for call forwarding.

How Supplementary Services Are Stored in BRM

A customer's supplementary services are defined in the service objects owned by the customer's account. For more information, see **/service/telco/imt and /service/telco/pdc objects**.

You define system-wide supplementary service definitions in **/config/telco/imt and /config/telco/pdc** objects. For example, supplementary IMT voice telephony services are defined in **/config/telco/imt/telephony object.** See "About Configuring Services, Supplementary Services, and ERAs".

About Extended Rating Attributes

Extended rating attributes (ERAs) provide discounts and promotions such as friends and family.

There are two types of ERAs:

- Account-level ERAs, for example a birthday discount, can be used with any type of service. Since account-level ERAs are not dependent on a service, account-level ERAs do not require any provisioning. If a customer account includes any IMT or PDC services, the account can use any account-level ERA.
- *Service-level ERAs*, for example, a closed user group, can only be used with a service. When a service is purchased, the account also gets all the service-level ERAs for that service. Service-level ERAs might require provisioning. If they do, they are implemented by using product-level provisioning.

ERAs need to be configured with specific data for each customer. For example, to set up a birthday discount, you need to know the customer's birthday. You configure ERAs by using your CSR tool.

Note:

- Even when an account is qualified to use ERAs, you do not have to implement them in the account.
- Sample ERAs are included in the default **pin_telco_tags_pdc_imt** and are included in the pipeline pricing data.
- BRM does not validate any data entered when configuring ERAs, for example, telephone numbers for the friends and family discount. To create validation rules for these entries, edit the PCM_OP_CUST_POL_PREP_PROFILE policy opcode.

You define ERAs in the pipeline **ISC_UsageType** module.

How ERAs Are Stored in BRM

A customer's ERA configurations are stored in profile objects.

- The **/profile/serv_extrating** object stores the service-level ERA configuration. These objects are linked to the service objects owned by the customer's account.
- The **/profile/acct_extrating** object stores the account-level ERA configuration. These objects are linked to account objects that own services.

You define system-wide definitions for service level ERAs in **/config/telco/***service_name* objects. For example, supplementary voice telephony services are defined in **/config/telco/***service_name***/telephony**. Account-level ERAs are defined in **/config/account_era** objects.

About Configuring Services, Supplementary Services, and ERAs

To configure IMT or PDC services, supplementary services, and ERAs, you define provisioning tags and account ERA definitions in the **pin_telco_tags_pdc_imt** file. You then run the **load_pin_telco_tags** utility to load the data into the BRM database. Provisioning tags and account ERA definitions are stored in service-specific **/config** objects, such as, **/config/telco/imt/telephony** and **/config/account_era**.

When you create your products in the BRM price list, you can assign different types of provisioning tags, for example, you can use a provisioning tag for:

- A single bearer service, for example, a type of voice service.
- One or more supplementary services without a bearer service. A product with this type of provisioning tag is typically included in an add-on plan, because the customer must have the service already before adding the supplementary services.
- One or more service-level ERAs. This type of provisioning tag can only be used in an add-on plan.
- Combinations of a bearer service, supplementary services, and service-level ERAs.

For example, you might include two different provisioning tags for a wireless telephony service product:

- The VoicePremium provisioning tag implements,
  - A voice bearer service.
  - The Call Forwarding supplementary service.

  - The Home Cell Assignment service ERA.
- The VoiceFamily provisioning tag implements,
  - A voice bearer service.
  - The Caller ID supplementary service.
  - The friends and family service ERA.

You might also create products like these:

- A product that implements a voice bearer service
- An add-on product that implements Call Forwarding and the Home Cell Assignment ERA
- An add-on product that implements Caller ID and the friends and family ERA

Important: You cannot directly change the status of supplementary services. Instead, you change their status by changing the status of the products that they were purchased with. For example, to inactivate a Call Forwarding supplementary service, you inactivate its product. However, when you do so, you inactivate all other products, supplementary services, and service-level ERAs that were purchased with that product. Therefore, you should create products that allow you to manage services after the products are purchased.

About Creating ERA Definitions and Provisioning Tags

You specify ERA definitions and provisioning tags by editing the **pin_telco_tags_pdc_imt** file.

The default set of ERAs is defined in the pipeline ISC_UsageType module. For a list of default ERAs, see "Default Service-Level ERAs".

Creating Custom ERAs

To create custom ERAs:

1. Define how the pipeline validates them by customizing the ISC_UsageType iScript.
2. Add the names and descriptions to the **era_descr.***locale* file.
3. Add the names to the **pin_telco_tags_pdc_imt** file, and load the file by using the **load_pin_telco_tags** utility. See "Loading Prerated Events" in *BRM Configuring Pipeline Rating and Discounting*.

Creating Account ERA Definitions

Use this syntax for an account-level ERA in the **pin_telco_tags_pdc_imt** file:

```
account_era "ERA code" "ERA name ID" "ERA description ID"
```

Example:

```
account_era "SPECIAL_DAY" "2" "3"
```

where:

- **account_era** specifies that this is an account ERA definition.

- **SPECIAL_DAY** is the ERA code. ERA codes are defined in the pipeline ISC_ UsageType module.
- **2** is the ERA name ID. This ID is defined in the **era_descr.***locale* localizable strings file.
- **3** is the ERA description ID. This ID is defined in the localizable strings file.

Creating Provisioning Tags

Provisioning tags are used for implementing bearer services, supplementary services, and service-level ERAs. This example shows a provisioning tag for a voice product that includes two supplementary services and a service-level ERA:

```
provisioning_tag  "/config/telco/IMT/voice"  "Voice"  "VoicePremium"  "Premium Voice Service"  "n"
service_era  "HOME_REGION"  14 15 "y"
```

Table 31–1 describes the provisioning tag syntax:

Table 31–1 Provisioning Tag Syntax

Tag Element	Description
provisioning_tag	Specifies the type of data (provisioning tag or account ERA). This entry includes the following values: ■ The object that stores the tag, for example: **"/config/telco/***service_name***/telephony"** The previous example specifies that this tag is stored in the **/config/telco/***service_name***/voice** object, for use with the IMT or PDC telephony service. ■ The service associated with the provisioning tag, for example: **"Voice"** ■ The name of the provisioning tag, for example: **"VoicePremium"** This name is displayed in Pricing Center. ■ The description of the provisioning tag, for example: **"Premium Voice Service"** By default, the description is not displayed in any application. ■ Unprovisioning requirement. **"y"** specifies that service cancellation triggers unprovisioning. **"n"** specifies that service cancellation does not trigger unprovisioning. You can use the unprovisioning requirement to leave a customer's service configuration unchanged. For example, you might want to leave a voice mailbox intact, in which case you would not unprovision the service. In most cases, you should trigger unprovisioning.

Table 31–1 (Cont.) Provisioning Tag Syntax

Tag Element	Description
service_era	Lists the service ERAs that are sold with the product, and specifies if provisioning is required, for example: **service_era "HOME_REGION" 14 15 "y"** See "Default Service-Level ERAs". The values in this line are: ■ **"HOME_REGION"** is the service ERA code. ERA codes are defined in the pipeline ISC_UsageType iScript. ■ **14** is the ID for the ERA name. The ID is defined in the **era_descr.***locale*localizable strings file. ■ **15** is the ID for the ERA description. The ID is defined in the localizable strings file. ■ **y** specifies that provisioning is required. **"n"** specifies that provisioning is not required. If the provisioning tag does not include a service-level ERA, leave this line out.

Loading Provisioning Tags

To load provisioning tags, you run the **load_pin_telco_tags** utility to load the contents of the **pin_telco_tags_pdc_imt** file into the BRM database.

Note: You cannot create provisioning tags and ERAs for individual brands. All provisioning tags and ERAs can be used by any brand.

For background information, see "About Supplementary Services".

Important: The **load_pin_telco_tags** utility requires a configuration file. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide*.

1. Edit the **pin_telco_tags_pdc_imt** file in *BRM_Home*/**sys/data/config**. The **pin_telco_tags_pdc_imt** file includes examples and instructions.

   Caution: By default, the **load_pin_telco_tags** utility overwrites the existing provisioning tags. If you are updating provisioning tags, you can use the **-x** option in the **load_pin_telco_tags** command to not delete tags that are not defined in the current **pin_telco_tags_pdc_imt** file.

2. Save the **pin_telco_tags_pdc_imt** file.
3. Run the **load_pin_telco_tags** utility:

   ```
   load_pin_telco_tags pin_telco_tags_pdc_imt
   ```

   For the complete command syntax, see **load_pin_telco_tags**.
4. Restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

5. Restart Pricing Center.

To verify that the **pin_telco_tags_pdc_imt** file was loaded, you can display the **/config** objects by using Object Browser, or use the **robj** command with the **testnap** utility. For example, to verify the provisioning tags for the telephony service, look at the **/config/telco/***service_name***/telephony** object.

See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.

Default Service-Level ERAs

Table 31–2 summarizes the default service-level ERAs:

Table 31–2 Default Service-Level ERAs

Function	Name	Pipeline Name
Create a closed user group, for example, a group of all mobile numbers in a company.	Closed user group ERA	CLOSEDUSERGROUP
Specify a discount account for calculating volume discounts across multiple accounts.	Hierarchical discount account ERA	DISCOUNTACCOUNT
Provide cross-product volume discounts, as defined in the pipeline.	Pipeline discount model ERA	DISCOUNTMODEL
Provide discounts to calls made to specific numbers, for example, friends and family.	Friends and family ERA	FRIENDS_FAMILY
Define the home cells for a customer and allow discounts while calling from this area.	Home cell assignment ERA	HOME_CELL
Provide discounts to calls made to specific numbers or regions, such as all numbers in a country or area code.	Home region code ERA	HOME_REGION
Use a pipeline rate plan.	Pipeline service-level rate plan ERA	RATEPLAN
Assign a Quality of Service to a service.	Service level agreement ERA	SERVICELEVEL

32

Adding a New IMT or PDC Service

This chapter provides information on how to extend the base IMT and PDC services that are provided by Oracle Communications Billing and Revenue Management (BRM) IMT Manager.

Before reading this section, read the following sections to understand the IMT Manager and install it:

- About Using IMT Manager
- Installing and Configuring IMT Manager

About Extending IMT and PDC Services

IMT Manager provides you with the base /service classes to rate and manage services that are provisioned on PDC and IMT networks. Before you can use IMT Manager, you must extend the base service and create groups of services that you offer to your customers, such as telephony, data, fax, and Web browsing, for example, **/service/telco/IMT/voice**.

You can offer multiple types of IMT and PDC services within each service group. For example, you can create a **service/telco/IMT/voice** class for normal voice service and emergency-only voice service.

Extending the IMT and PDC Services

To add a new service to the base IMT and PDC services:

1. Create a subclass of the IMT and PDC /service class by using Developer Center. For example, **/service/telco/IMT/voice**.

   See "Adding Support for a New Service" in *BRM Developer's Guide*.

2. Customize the **fm_subscription_pol_provisioning.c** file in the *BRM_Home*/**source/sys/fm_subscription_pol** directory to implement the functions that process the service data.

   See "Customizing the Policy FM for a New IMT or PDC Service".

3. Modify the Rated Event (RE) Loader control files and then configure them.

   See "Creating Control Files for a New IMT or PDC Service".

4. Edit the **pin_telco_tags_pdc_imt** file to configure services, supplementary services, and ERAs, and then load the file into the BRM database.

   See "Configuring IMT and PDC Services and Extended Rating Attributes".

5. Edit the **pin_event_map** file to map services to event types for each new service.

See "Mapping Event Types to Services".

6. To specify bill items specific to your IMT services, edit and load the **config_items_tags_imt** and **config_items_types_imt** files.

   See "Configuring Bill Items".

7. To specify bill items specific to your PDC services, edit and load the **config_items_tags_pdc** and **config_items_types_pdc** files.

   See "Configuring Bill Items".

8. Add entries to the **ifw_service** and **ifw_ref_map** files to configure the pipeline for the new service and event types.

   See "Configuring the Pipeline for Services and Event Types".

9. For each service, customize the following opcodes to implement your business policy:
   - PCM_OP_IMT_POL_APPLY_PARAMETER
   - PCM_OP_PDC_POL_APPLY_PARAMETER
   - PCM_OP_SUBSCRIPTION_POL_GET_PROD_PROVISIONING_TAGS

10. Create rate plans for your new services.

    See "Setting Up the IMT and PDC Price Lists".

Customizing the Policy FM for a New IMT or PDC Service

To customize the policy FM for a new service:

1. Open the **fm_subscriptions_pol_provisioning.c** file in the *BRM_Home*/**source/sys/fm_subscription_pol** directory.

2. Define the macros for the configuration, service, and service profile objects.

   For example, to add a new IMT voice service, add these definitions:

```
#define TELCO_PDC_VOICE_CONFIG            "/config/telco/imt/voice"
#define TELCO_PDC_VOICE_SERVICE           "/service/telco/imt/voice"
#define TELCO_PDC_VOICE_SERVICE_PROFILE   "/service/telco/imt/voice"
```

3. Implement the function for collecting provisioning tags.

   For example, to add a new IMT voice service, add this code:

```
static void tag_fn_imt_voice_tcf(pcm_context_t *ctxp, poid_t   *svc_obj_p,pin_
flist_t *r_flp, int32 *eidp, pin_errbuf_t *ebufp)
{
   tag_fn_telco(ctxp, svc_obj_p, r_flp, eidp, ebufp,
   TELCO_IMT_VOICE_CONFIG, TELCO_IMT_VOICE_SERVICE_PROFILE);
   PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG,"tag_fn_pdc_voice_tcf return flist",r_
flp);
}
```

4. Implement the function to validate the service tag.

   For example, to add a new IMT voice service, add this code:

```
static int32 valid_tag_imt_voice_tcf(pcm_context_t *ctxp,
                     poid_t *svc_obj_p, char *tag)
{
  return (int32)valid_tag_tcf_telco(ctxp, svc_obj_p, tag, TELCO_IMT_VOICE_
CONFIG);
```

```
}
```

5. Add a new entry for the service in the **service_info** table.

   For example, to add a new IMT voice service, add this structure:

```
static struct service_info
      {
         char  *service;
         void  (*plp_func)(pcm_context_t *, poid_t *, int32, char pin_errbuf_t
*);
         void  (*tag_func)(pcm_context_t *, poid_t *, pin_flist_t *,int32 *,
pin_errbuf_t *);
           int32 (*valid_func)(pcm_context_t *, poid_t *, char *);
      }
        service_info[] =
        {
           ……
        {
          TELCO_IMT_VOICE_SERVICE, plp_imt_tcf, tag_fn_imt_voice_tcf,
          valid_tag_imt_voice_tcf
        },
         {NULL,NULL,NULL,NULL}
      };
```

6. Save, close, and compile the file.

Creating Control Files for a New IMT or PDC Service

The control files represent the extended event structure table of each event type. For each new service you create, you need to provide a control file. RE Loader uses the control file to map and load the extended event structure table.

For more information about RE Loader, see "Configuring Rated Event Loader" in *BRM Configuring Pipeline Rating and Discounting*.

IMT Manager includes the following control files for the base IMT and PDC services:

- **event_session_imt.ctl**
- **event_session_pdc.ctl**

When you add new services, you must create a control file for each service and add the required column in the files for the service.

> **Important:** The examples show a new IMT voice service. All the values and strings are only examples and must be replaced with values for your services.

To load the extended event structure into the BRM database:

1. Open the **Infranet.Properties** file in the *BRM_Home*/**apps/pin_rel/***Service_name* directory and add the newly created table name.

   For example:

```
infranet.rel.storable_class.event_session_telco_imt.table.5.name = event_
session_telco_imt_voice_t
```

2. In the *BRM_Home*/**apps/pin_rel/***Service_name* directory, create a control file for each new service, for example, **event_session_imt_voice_t.ctl**.

Note: You can copy the **event_session_imt.ctl** file to the same directory and make the changes by following the instructions in the file.

3. Edit the **pin_rel_preprocessor_cdr.pl** file in the *BRM_Home*/**apps/pin_rel/***Service_name* directory:

   a. Add an entry for each new service under the section for constants required for CDR files. For example, for a new voice service, enter:

   ```
   $voiceRec ="650";
   ```

   b. Add an entry for the new service under the section for construct output file names. For example:

   ```
   elsif ($table[$i] =~ /^event_session_tlco_pdc_voice_t$/i)
   {
     $outFile8 = $outFile;
     print "Output file 8 : $outFile8\n" if $debug;
   }
   ```

   c. Add an entry for opening remittance under section for opening the input file:

   ```
   if (defined $outFile8)
   {
      open(OUT8_FILE, ">$outFile8") || exit_err ($CANT_   OPEN  FILE,
   $outFile8, $!);
   }
   ```

   d. Modify the code:

   ```
   elsif(($line =~ m/^$gprsRec/o) || ($line =~ m/^$gsmRec/o) || ($line =~
   m/^$voiceRec/o))
   {
     if( $foundRecForFile2 == 1) {
     $foundRecForFile2 = 0;
     print OUT5_FILE "$line$deli\n";
     print OUT7_FILE "$line$deli\n";
     # Appending the event poid to the cp details record for   # REL loading
     printf(OUT8_FILE "$line$deli$high%07u\n",$i);
   }
   ```

   e. Add code for closing the file at the end of file.

   ```
   if (defined $OUT8_FILE) {
   close(OUT8_FILE);
   }
   ```

   f. Save and close the file.

Mapping Event Types to Services

To specify which events to rate and to map the event types to services, you define the mapping and load it into the **/config/event_map** object in the BRM database.

1. Open the **pin_event_map** file in the *BRM_Home*/**sys/data/config** directory.

   This file includes entries for the base services and instructions for adding new entries.

2. Add entries for the new service and the events to rate for the service.

3. Save and close the file.
4. Run the **load_event_map** utility by using this command:

```
load_event_map pin_event_map
```

> **Caution:** The **load_event_map** utility overwrites the entire event map. If you are updating the event map, you cannot load new mappings only. You must load the entire event map each time you run the **load_event_map** utility.

For more information, see "load_event_map" in *BRM Setting Up Pricing and Rating*.

Configuring Bill Items

To provide service-specific bill items on your invoices for your IMT and PDC services:

1. Create a **config_items_tag**_*service_name*.**xml** and a **config_items_type**_*service_name*.**xml** file with data relevant to the service. For example, **config_item_tags_pdc.xml** and **config_item_types_pdc.xml.**

   Use the **config_item_tags.xml** and **config_item_types.xml** files in the *BRM_Home*/**sys/data/pricing/example** directory as examples.

   For more information, see "Creating Custom Bill Items" and "Assigning Items Tags Based on Event Attributes" in *BRM Configuring and Running Billing*.

2. Load the files you created into the **/config/item_tags** and **/config/item_types** objects in the BRM database:

```
load_config_item_tags [config_item_tag_service_name]
load_config_item_types [config_item_type_service_name]
```

For more information, see "load_config_item_tags" and "load_config_item_types" in *BRM Configuring and Running Billing*.

Configuring the Pipeline for Services and Event Types

To enable the pipeline to process EDRs, you need to map external service codes to internal service codes, /service classes, usage classes, and usage types. IMT Manager includes sample database configuration files for mapping the base IMT and PDC services. When you add new services, you need to create service mappings by using Pricing Center.

For more information, see the following topics in the BRM documentation:

- About mapping services
- Mapping service codes and /service classes
- Mapping events and services

Setting Up the IMT and PDC Price Lists

Use Pricing Center to set up rate plans for your IMT and PDC services. Use the sample GSM price list as an example when creating price lists for your services.

See "Setting Up GSM Wireless Pricing".

Instead of using Pricing Center to create and load your price lists, you can also use the XML pricing interface to create the price lists and use the **loadpricelist** utility to load

the price list into the database. For more information, see "loadpricelist" and "Using the XML Pricing Interface to Create a Price List" in *BRM Setting Up Pricing and Rating*.

End-to-End Testing Your IMT Manager Implementation

When you have installed all of the required components and have made sure that each is installed correctly and functioning, you can perform an end-to-end test of the entire wireless implementation.

1. Load your price lists into BRM.
2. Load the pricing data in the pipeline.

   See "Setting Up Pipeline Price List Data" in *BRM Setting Up Pricing and Rating* and "Installing Pipeline Manager" in *BRM Installation Guide*.
3. Create an account and rate a usage event for that account
4. To rate a usage event, create a sample CDR file that uses the test account's phone number as the originating number.

   Important: In your sample CDR, the event dates must be later than the account creation dates.

By creating an account, you ensure the following:

- Services and ERA provisioning tags have been configured.
- The price list has been loaded.
- The Account Synchronization Data Manager and the Pipeline Listener can update the pipeline with a new account.

By rating a usage event, you ensure the following:

- The pipeline is configured correctly.
- The pipeline can get data from the BRM database.
- RE Loader is configured correctly.

Part VII

Managing Telephone Number Inventory

Part VII describes how to manage telephone number inventory in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

- About Managing Telephone Numbers
- Installing and Configuring Number Manager and Number Administration Center
- Number Manager Utilities

33

About Managing Telephone Numbers

This chapter describes how to manage telephone number inventory in your Oracle Communications Billing and Revenue Management (BRM) system by using Number Manager and Number Administration Center.

You need to manage number inventory when you offer GSM services. For information, see "About Integrating Wireless Services".

Before setting up number management, you need to know the following information:

- Basic BRM concepts. See *BRM Concepts*.
- How to create and edit BRM configuration files. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Important: Number Manager and Number Administration Center are optional components.

About Managing Numbers

To manage telephone numbers, you use the following components:

- *Number Manager* includes the number management opcodes, utilities, configuration files, and storable classes.
- *Number Administration Center* is a GUI application used by operations personnel to manage telephone number inventory.

You use Customer Center to assign and manage numbers in customer accounts. For more information, see information about working with GSM accounts in the Customer Center Help.

About Using Number Administration Center

To create an inventory of telephone numbers in the BRM database, you use Number Administration Center to create blocks of numbers. Blocks of numbers usually correspond to geographic regions or business entities. When you create a block of numbers, all the numbers in the block share the block name and the telephone number attributes, such as the network element (also known as home location register, or HLR), and brand. See "Creating a Block of Telephone Numbers".

About Managing Blocks and Numbers

After creating the numbers as devices in the BRM database, you can do the following:

- Modify a block. You can change the block name, brand, and prefix (for example, the North American area code) of all the numbers in the block. See "About Modifying a Block".
- Change the size of an existing number block. See "Changing the Size of the Number Block".
- Split a block. You can split a block into smaller blocks, but you cannot join blocks. See "Splitting a Block".
- Change telephone number attributes for one or more numbers. See "Updating Telephone Numbers".
- Manage how numbers are quarantined after a customer cancels a service. See "About Number Device States".

About Managing Numbers with Branded Accounts

When you assign a brand to a block, the telephone numbers in that block are available only to that brand. When you choose a number in Customer Center, you can only choose numbers in the brands for which you have permissions.

In Number Administration Center, brand access is determined by the login. You can create and modify blocks of numbers only for the brands you have permission to access. (For information about brand access, see "About Granting Access to Brands" in *BRM Managing Customers*.)

When you create a block, you can assign it to a brand, but you can change a block's brand after you create it. However, you cannot change the brand of individual telephone numbers.

If you have access to multiple brands, you can search for blocks or telephone numbers by choosing the brand name.

See "About Branding" in *BRM Managing Customers*.

Note: If you use a multischema system, see "Managing Numbers in a Multischema System" for more information about using brands.

Brand-Aware Number Attributes

You can configure the following number attributes for each brand:

- You can define brand-specific device states. See "Device Management and Brands" in *BRM Developer's Guide*.
- You can set the default quarantine period for each brand. See "Changing the Default Quarantine Period".

Number Attributes that Are Not Brand Aware

You cannot configure the following number attributes for specific brands. Instead, they apply to all numbers in the BRM system:

- Number categories
- Network elements
- Vanity types

Managing Numbers in a Multischema System

If you manage numbers in a multischema system:

- You cannot change the brand of a block to a brand that is hosted in a different database schema from the schema that the current brand is hosted in. For example, if a block belonging to Brand A is in schema 0.0.0.1, you cannot change the block to Brand B if Brand B is in schema 0.0.0.2.
- If you log in as a brand host administrator and select Brand Host as the brand to create a new block, you can create blocks only in the schema that you logged in to. If you select any other brand, the block is created in the schema that hosts the selected brand.
- When searching for blocks or numbers, if you log in as a brand host administrator and you select the top-level brand without including sub-brands in the search criteria, the search results include only numbers or blocks from the schema you logged in to.
- In a multischema environment that is not brand enabled, you can create numbers only in the schema you are logged in to.

Getting Information about Number Usage

You can get information about number inventory and status by running the Number Manager reports. See "Number Manager Reports" in *BRM Reports*.

In addition, you can display the history of a customer's number in Customer Center. See information about displaying telephone number history in the Customer Center Help.

You can also display the number inventory in Number Administration Center. See "Displaying the Number Inventory".

About Managing Telephone Numbers in Customer Center

To create accounts and manage telephone numbers in existing accounts, use Customer Center. You can perform the following tasks:

- Assign telephone numbers to services.
- Change a customer's telephone number.
- Display the history of a customer's number.

For more information, see information about working with GSM accounts in the Customer Center Help.

About Number Device States

Telephone numbers have the following device states: new, assigned, quarantined, and unassigned.

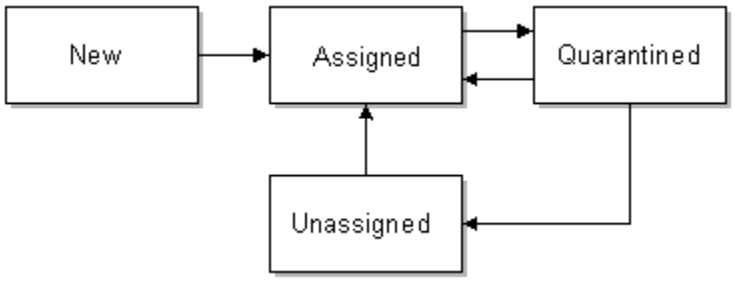
- When you create a number, the device state is *New*.
- When you assign a number to a customer, the device state is *Assigned*.
- When a customer cancels the service associated with the number, the device state is *Quarantined*.

Note: By default, you can assign a quarantined number to an account. You can modify the PCM_OP_NUM_POL_DEVICE_ASSOCIATE policy opcode to not allow assigning a quarantined number to an account. You can also write a custom opcode and use it in the device state transition from quarantined to assigned. See "Customizing Device States".

- By default, a number is unquarantined after 90 days. After a number has been unquarantined, the device state changes to *Unassigned*.

Figure 33–1 shows how number device state changes:

Figure 33–1 Number Device State Changes



You cannot use Number Administrator to change the number device state. Instead, the number device state is changed automatically when the number is assigned to an account and when an account or service changes status (for example, when a customer cancels the service that the number is associated with).

Note: You can create your own number device states. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

Taking Numbers Out of Quarantine

By default, a number is quarantined for 90 days. You can change the number of days by running the "load_pin_num_config" utility. See "Changing the Default Quarantine Period".

If you run out of new or unassigned numbers, you can use the "pin_change_num_quarantine" utility to unquarantine numbers. See "Changing the Quarantine Status Manually".

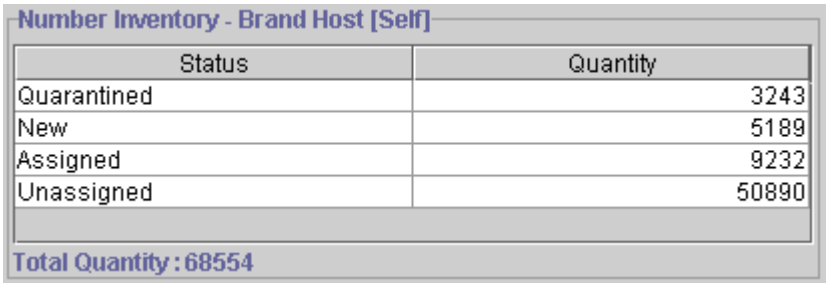
Displaying the Number Inventory

By default, Number Administration Center displays the number inventory when you start the application. To display it at any time, choose **View - Number Inventory**.

If you log in as a brand account, the inventory is shown for your brand.

Figure 33–2 shows the Number Inventory window.

Figure 33–2 Number Inventory Window



You can also display the number status of the numbers in a single block. See "About Modifying a Block".

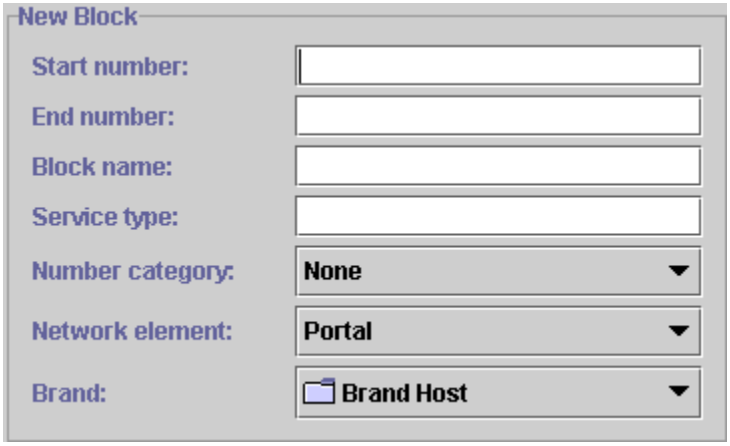
Creating a Block of Telephone Numbers

Important: Before creating numbers, you must define network elements and any other number properties required by your system (for example, number categories). See "Mandatory Configuration Tasks".

To create a block of numbers in the BRM database, start Number Administration Center and choose **File - New Block**.

Figure 33–3 shows the block creation screen in Number Administration Center:

Figure 33–3 Block Creation in Number Administration Center



To create a block of numbers, you enter the following information:

- A start number and an end number that together form the range of numbers in the block. You can use any common format when entering numbers. For example:
  - 1-800-555-1212
  - 1(800) 555-1212
  - 1.800.555.1212
  - 18005551212

- A block name. This can be any name you want, but you can only use the same name for one block. The maximum length is 255 characters. You can change the block name at any time.

In addition to the block name and the range of numbers in the block, you assign the following number attributes to all the numbers in the block:

- The *brand*. When you assign a brand to a block, the telephone numbers in that block are available only for that brand. When you choose a number in Customer Center, you can only choose numbers in the brands for which you have permissions.

  When you assign the brand, you can choose only from the brands for which you have access. If you have the proper brand permissions, you can change the brand after you create the block.

- The *network element*. You can change the network element after you create the block.

  To specify the network element, you choose from a list. By default, there are no network elements; you must add your own. You can also localize network elements. See "Creating Network Elements".

- The *number category*. This is a configurable field that you can use for managing numbers. It is typically used for identifying corporate entities, but it can be used for defining any type of number classification. For example, you can create number categories for business or residential numbers. You can also localize number categories.

  You can change the number category after you create the block.

  To customize the list of number categories, see "Creating Number Categories". By default, the number categories are *None* and *Reserved*.

- The *vanity type*. This attribute can be used for customized pricing and rating (see "Rating Implementation and Customization" in *BRM Setting Up Pricing and Rating*). Number Manager includes only sample vanity types; if you use vanity types, you must create your own. You can localize vanity types. See "Creating Vanity Types".

  When you create a block of numbers, you cannot choose the vanity type for the entire block. By default, the vanity type applied to a new block is the first vanity type defined in the **num_vanities.***locale* file. After creating the numbers, you can apply vanity types to numbers.

- The *service type*. You can use the service type to customize business policies. By default, the service type is not applied to any business logic. See "Customizing How Service Types Are Used".

About Customizing Block Creation and Modification

By default, BRM prevents the creation of duplicate numbers. To customize how numbers are created (for example, to prevent numbers or ranges of numbers from being created), edit the PCM_OP_NUM_POL_DEVICE_CREATE source code.

- You can change the prefix (for example, the North American area code) for a block of numbers. To allow changing of other parts of the number (for example, country codes), modify the PCM_OP_NUM_POL_DEVICE_SET_ATTR source code.

About Modifying a Block

To modify a block:

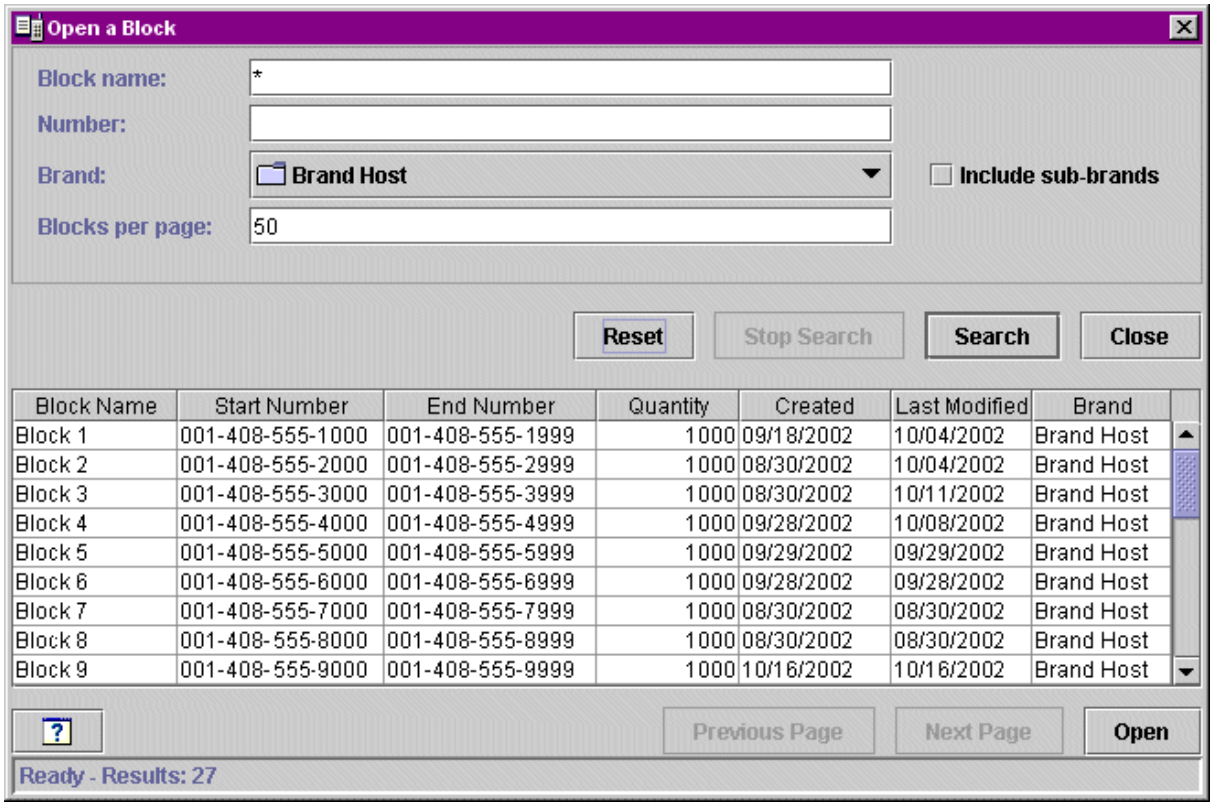
1. In Number Administration Center, choose **File - Open a Block**.
2. Enter the search criteria. You can search for block name, telephone number, and brand. You can use wildcards when searching for numbers:
   - Use the asterisk (*) to match zero or more characters.
   - Use the question mark (?) to match a single character.
   - To search by brand, choose the brand from the list. To search in all of the brand's sub-brands, select **Include sub-brands**.

     If you do not use a brand-enabled system, the **Brand** search attribute and the **Brand** column are not shown.

   The search results display the block and show the block name, start and end numbers, quantity of the range, brand, when the block was created, and when it was last modified.

   Figure 33–4 shows the Open a Block dialog box:

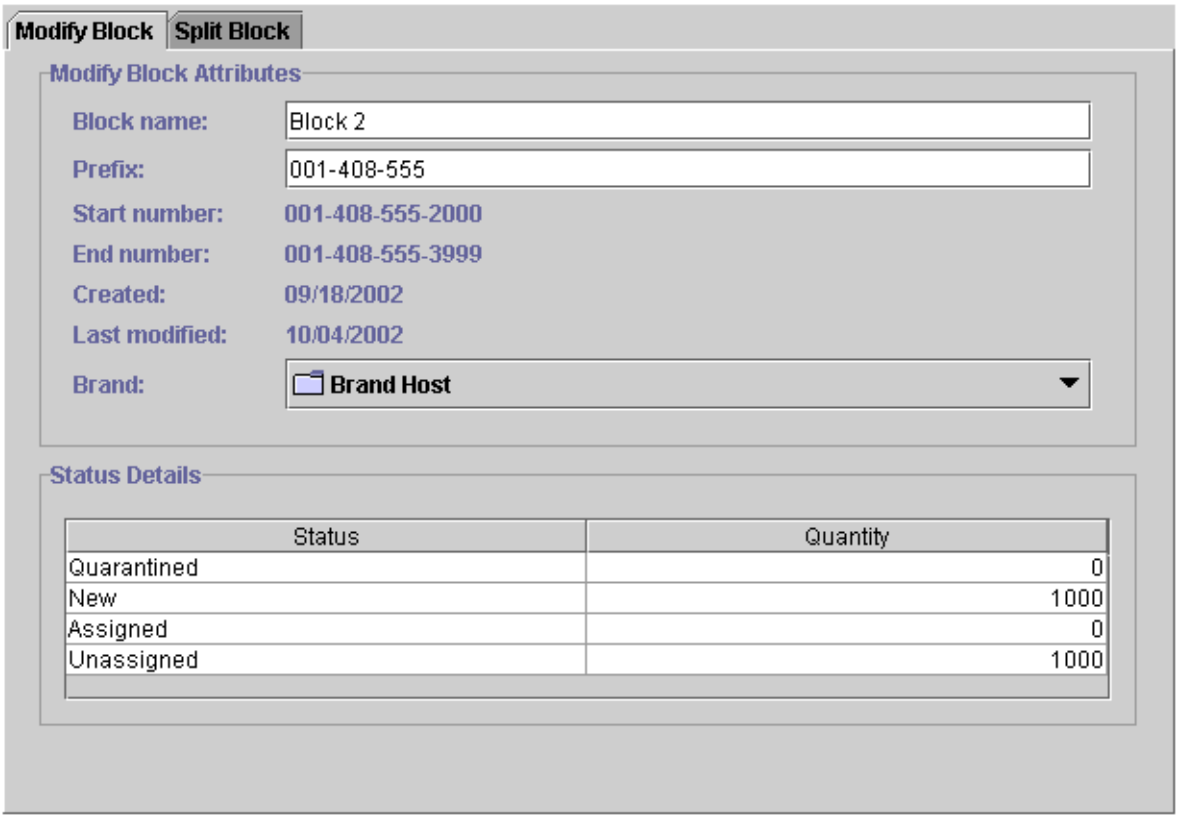
Figure 33–4 Open a Block Dialog Box



3. Select the block and click **Open**.

   Figure 33–5 shows the **Modify Block** tab:

Figure 33–5 Modify Block Tab



4. Change the block name, prefix, or brand. By default, you can change only the North American area code digits.
5. Choose **File - Commit to Database**.

Changing the Size of the Number Block

You can extend the block size by specifying a new higher value for the end number. You can shrink the range of an existing block of numbers by doing either of the following:

- Specifying a new lower value for the end number.
- Specifying a new higher value for the start number.

To change the size of the number block:

1. In Number Administration Center, choose **File - Open a Block**.
2. Enter the search criteria.

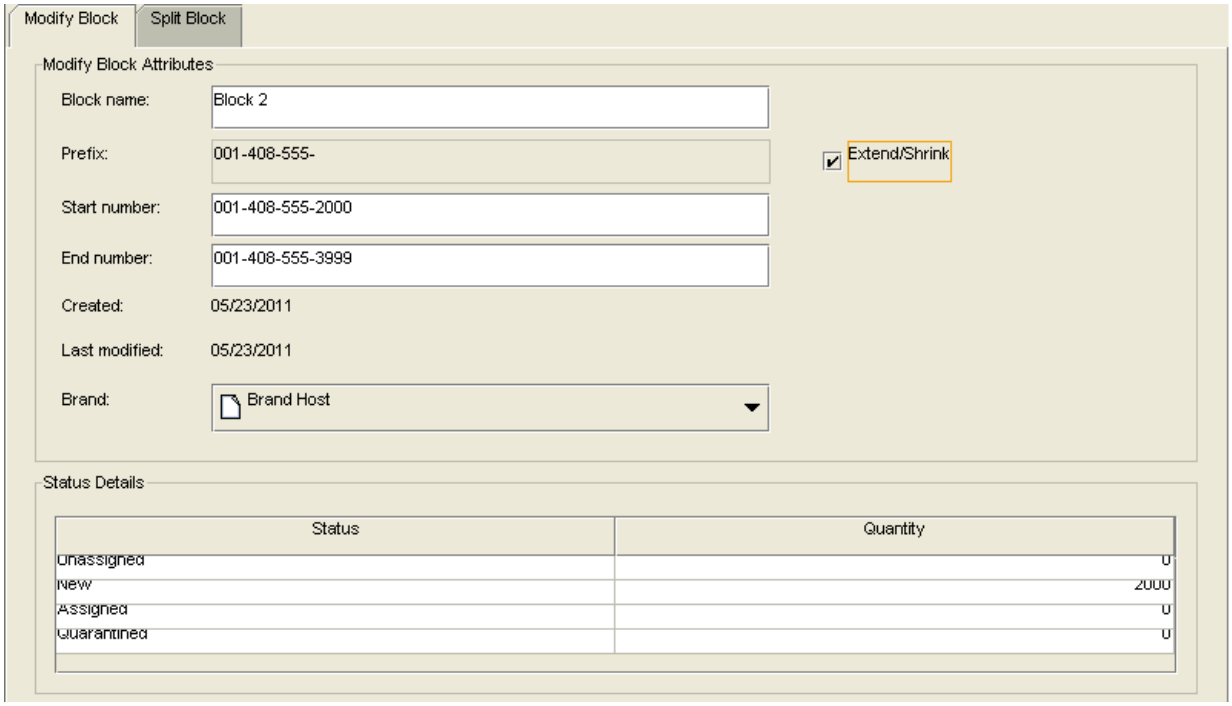
   You can search for block name, telephone number, and brand. You can use wildcards when searching for numbers:

   - Use the asterisk (*) to match zero or more characters.
   - Use the question mark (?) to match a single character.
   - To search by brand, choose the brand from the list. To search in all of the brand's sub-brands, select **Include sub-brands**.

If you do not use a brand-enabled system, the **Brand** search attribute and the **Brand** column are not shown. The search results display the blocks and show the block name, start and end numbers, quantity of the range, brand, when the block was created, and when it was last modified.

3. Select the block and click **Open**.
4. In the **Modify Block** tab, select the **Extend/Shrink** check box as shown in Figure 33–6.

Figure 33–6 Extend/Shrink Check Box



5. Do one of the following:
   - To extend the number block, in the **End number** field, enter a value that is higher than the previous end number.

     Note: The new end number that you specify must be in a continuous range and not overlap with any other existing blocks.

   - To shrink the number block, in the **End Number** field, enter a value that is less than the previous end number. Alternatively, in the **Start Number** field, enter a value that is higher than the previous start number.
6. Choose **File - Commit to Database**.

Splitting a Block

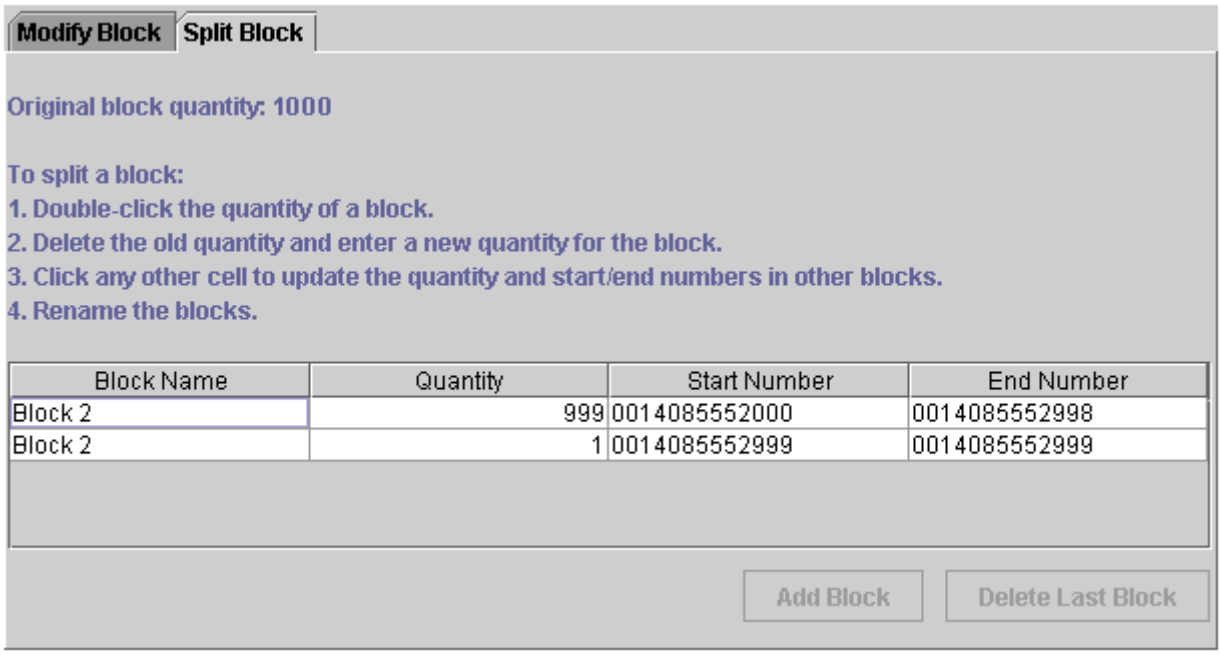
You can split a block into two or more blocks. After you split a block, you can modify the blocks (for example, change the North American area code) or brand of the numbers in the blocks.

1. In Number Administration Center, choose **File - Open the block**.

2. Click the **Split Block** tab.

   By default, two rows are displayed in a table as shown in Figure 33–7:

Figure 33–7 Splitting a Block



   You can use two methods for splitting a block:

   - Enter a quantity in either of the rows. It must be smaller than the current quantity of the block.
   - Enter an end number. It must be a number within the current block range. You cannot enter a start number.

     In either case, the quantity and start and end numbers are updated to match the new block size.

3. Enter names for the new blocks.
4. Click **Add Block**.
5. Click **Commit**.

Updating Telephone Numbers

You can use Number Administration Center to search for telephone numbers and modify them. You can update all number attributes except the status and the number itself. For information about number attributes, see "Creating a Block of Telephone Numbers".

Note:

- You can update numbers by updating a block of numbers. For example, you can change the prefix (by default, the North American area code) for a block of numbers.
- Changing a telephone number or changing any of its attributes does not create a new **/device/number** object in the database. Instead, the existing number is updated.
- It is possible to delete numbers by using opcodes, but it is not recommended. You could reduce data integrity, or accidentally delete a number that is in use.

To update a number:

1. In Number Administration Center, choose **File - Search Numbers**.

   The Search window opens.

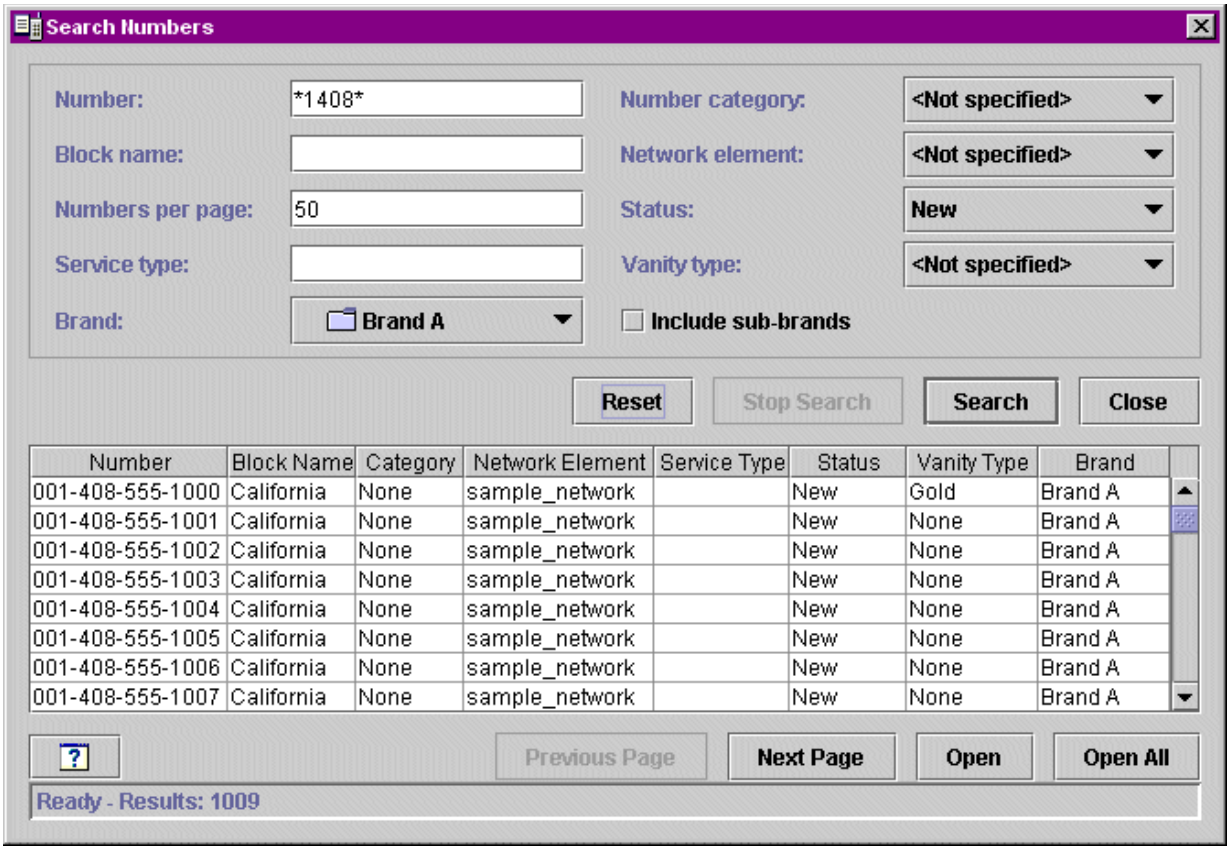
2. Enter the search criteria.

   You can search on any number attribute. You can use wildcards when searching for numbers and number category:

   - Use the asterisk (*) to match zero or more characters.
   - Use the question mark (?) to match a single character.
   - Number Administration Center ignores all other punctuation. For example, searching for ***8-5*** is the same as searching for ***85***.

   Figure 33–8 shows the Search Numbers dialog after a search:

Figure 33–8 Search Number Dialog Box



3. Open the numbers that you want to modify.
   - To open all numbers found in the search, click **Open All**.
   - To open the page of numbers that is currently displayed, click **Open**. Click **Next Page** and **Previous Page** to display different numbers.
   - Select one or more numbers and click **Open**.
   - Choose **File - Search Numbers** to return to the search results.

     To select numbers:

     – To select one number, click it.

     – To select all numbers, click **Select All**.

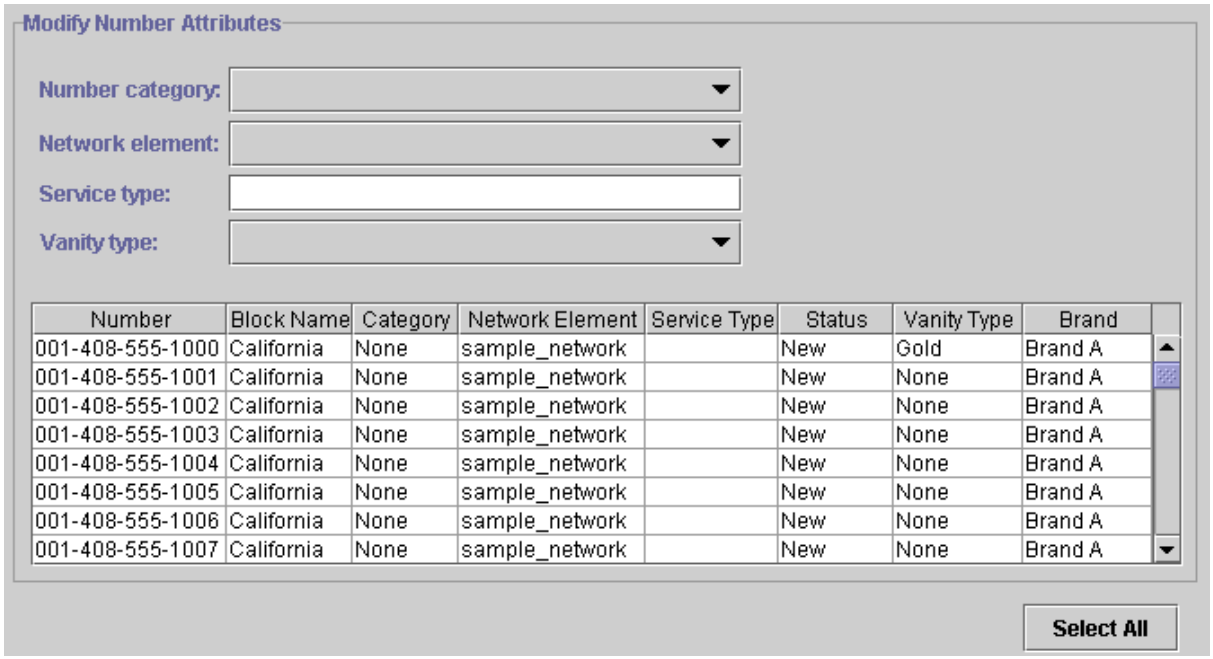
     – To select a range of numbers, press **SHIFT** and click the first and last numbers in the range.

     – To select multiple numbers not in a range, press **CTRL** and click the numbers.

Note: If the quantity of numbers found is the same or less than the quantity of numbers per page, the **Open** and **Open All** buttons open the same set of numbers. For example, if you specify 100 numbers per page, and only 50 numbers are found, both buttons open 50 numbers.

Figure 33–9 shows the Modify Number Parameters window:

Figure 33–9 Modify Number Parameters Window



4. Select the numbers that you want to modify.
5. Enter the new number attributes.
6. Click **Commit**.

About Number Customization Options

You can customize how you manage number inventory as follows:

- You can customize network elements.

  Important: By default, there are only sample network elements; you must add your own. See "Creating Network Elements".

- You can customize number device and service associations. See "Customizing Number Device Service Associations".
- You can customize number device states. See "Customizing Device States".
- You can customize how numbers are quarantined. See "Customizing Number Quarantine".
- You can create number categories. See "Creating Number Categories".
- You can create vanity types. See "Creating Vanity Types".
- You can customize how service types are used. See "Customizing How Service Types Are Used".
- You can customize how Number Administration Center works. See the following:
  - Changing the Number Administration Center Number Display Format
  - Configuring Search Performance in Number Administration Center

About the Number Manager Opcodes

To create telephone numbers in the BRM database, Number Manager opcodes do the following tasks:

- Create and modify blocks of numbers.
- Manage number quarantine.
- Manage number portability.

Creating Blocks of Numbers

To create a block of numbers, use the PCM_OP_NUM_CREATE_BLOCK opcode.

PCM_OP_NUM_CREATE_BLOCK calls the PCM_OP_CUST_POL_CANONICALIZE and the PCM_OP_NUM_POL_CANONICALIZE policy opcodes to format the block name and telephone numbers. It calls the PCM_OP_DEVICE_CREATE opcode to create numbers and the PCM_OP_CREATE_OBJ opcode to create the block.

This opcode returns an error if the block name already exists or if any of the telephone numbers already exists.

Modifying Blocks of Numbers

To modify a block of numbers, use the PCM_OP_NUM_MODIFY_BLOCK opcode.

If numbers are changed, this opcode calls the PCM_OP_NUM_POL_CANONICALIZE policy opcode to normalize the number and the PCM_OP_DEVICE_SET_ATTR opcode to change attributes such as the manufacturer, model number, and description.

This opcode returns an error if the new block name already exists or if the range of numbers specified is not valid.

If this opcode runs successfully, it returns the POID of the modified block and an array of POIDs for new blocks.

Splitting Blocks of Numbers

To split a block of numbers, use the PCM_OP_NUM_SPLIT_BLOCK opcode.

The input includes the POID and name of the existing block and the start and end numbers for each block.

If this opcode runs successfully, it returns the POIDs of the original block and the new blocks.

If an invalid block or number attribute is found, this opcode returns an error to the error buffer.

Managing Number Quarantine

To quarantine numbers, use the PCM_OP_NUM_QUARANTINE opcode.

This opcode creates or deletes a **/schedule/device** object to manage the telephone number quarantine. PCM_OP_NUM_QUARANTINE is called by the PCM_OP_DEVICE_SET_STATE opcode when the number device state changes in and out of the Quarantined state. This is the default behavior as specified in the number device state configuration.

- If the state transition is to Quarantined, this opcode calls the PCM_OP_ACT_SCHEDULE_CREATE opcode to create a **/schedule/device** object that, at the end of the quarantine period, will be used to change the state to Unassigned.
- There are two possible exit transitions from the Quarantined state:
  - If the state transition is from Quarantined to Assigned, this opcode calls the PCM_OP_SEARCH opcode to find the **/schedule/device** object associated with the number and calls the PCM_OP_ACT_SCHEDULE_DELETE opcode to delete the **/schedule/device** object.
  - If the state transition is from Quarantined to Unassigned, this opcode calls the PCM_OP_DEVICE_SET_ATTR opcode to delete the text in the device description field so it does not display in Customer Center.

To customize how number quarantine works, you can edit and load the **pin_device_state** file to call a custom opcode instead of calling PCM_OP_NUM_QUARANTINE. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

Managing Number Portability

To manage number portability, use the PCM_OP_NUM_PORT_IN and PCM_OP_NUM_PORT_OUT opcodes.

PCM_OP_NUM_PORT_IN creates a number device using the provided number. This opcode is used when porting the number from another service provider into the existing network.

PCM_OP_NUM_PORT_IN calls standard opcodes to do the following:

- Create the number and the **/device/num** object.
- Set the status of the **/device/num** object.
- Reformats the number provided in the input flist to your required format.
- Validate the required information in the input flist.
- Search for the provided number in the database and, if found, return an error.
- If not found, create the number in the number device and update the status of the number device and commit the number to associate the device with the account.

PCM_OP_NUM_PORT_OUT sets the status of the specified telephone number as **quarantine_port_out** in the **/device/num** object. PCM_OP_NUM_PORT_OUT is used when porting the number from your network to another service provider.

> **Note:** The present state of the device must be assigned and the new state must be set to **quarantine_port_out**. If these are not the states, PCM_OP_NUM_PORT_IN returns an error.

PCM_OP_NUM_PORT_OUT calls standard opcodes to perform the necessary validations and set the status of the number device. This opcode:

- Validates the number to be ported out.
- Validates the current status of the number device.
- Updates the status of the number device.

Customizing Number Manager

Use the following policy opcodes to customize Number Manager:

- PCM_OP_NUM_POL_CANONICALIZE. See "Customizing Number Normalization".
- PCM_OP_NUM_POL_DEVICE_ASSOCIATE. See "Customizing How Numbers Are Associated with Services".
- PCM_OP_NUM_POL_DEVICE_CREATE. See "Customizing Telephone Number Attributes".
- PCM_OP_NUM_POL_DEVICE_SET_ATTR. See "Customizing How a Number Can Be Changed".
- PCM_OP_NUM_POL_DEVICE_SET_BRAND. See "Changing a Block's Brand".

Customizing Number Normalization

To customize number normalization, use the PCM_OP_NUM_POL_CANONICALIZE policy opcode. This policy opcode handles number normalization when receiving numbers from applications and outputting numbers to other opcodes or applications.

The PCM_OP_NUM_POL_CANONICALIZE opcode is called by PCM_OP_NUM_CREATE_BLOCK opcode when a block of numbers is created.

You can customize this policy opcode to change normalization rules for handling numbers.

By default, this policy opcode performs the following translations:

- Adds two leading zero characters (00) if they are missing from the input.
- Removes all characters except digits.

The PCM_OP_NUM_POL_CANONICALIZE policy opcode returns an error when the minimum or maximum length is not valid. These values are defined in the PIN_NUM_CANON_MIN_LENGTH and PIN_NUM_CANON_MAX_LENGTH entries in the **pin_num.h** file in *BRM_Home*/**include**.

If successful, this policy opcode returns the normalized string.

Customizing How Numbers Are Associated with Services

To customize how numbers are associated with services, use the PCM_OP_NUM_POL_DEVICE_ASSOCIATE policy opcode. This policy opcode specifies the rules for associating or disassociate a number and a service. This policy opcode is called by the PCM_OP_DEVICE_ASSOCIATE opcode when a number is associated or disassociated with a service.

- If the number is being associated and there is no service currently associated with the number, this policy opcode calls the PCM_OP_DEVICE_SET_STATE opcode to change the state to Assigned. This is the default behavior as specified in the number device state configuration.
- If the number is being disassociated, and there is only one service associated with the number, this policy opcode calls the PCM_OP_DEVICE_SET_STATE opcode to change the state to Quarantined. This is the default behavior as specified in the number device state configuration.

When associating a number that is already associated with a service, the following rules are applied:

- A number can be associated with multiple services, but each service must be of a different type. For example, you cannot associate one number with two SMS services. To do so, you need to customize the PCM_OP_NUM_POL_DEVICE_ASSOCIATE policy opcode.
- A number can be associated with multiple services, but all services must belong to the same account.
- If an associated service has a SIM card device associated with it, the number and the SIM card must have the same network element.

To allow Pipeline Manager to find an account based on the IMSI or MSISDN in the CDR, the PCM_OP_NUM_POL_DEVICE_ASSOCIATE policy opcode copies the IMSI and MSISDN to the alias list array in the associated service object. The element ID is significant:

- The IMSI is copied to ALIAS_LIST[0]
- The MSISDN is copied to ALIAS_LIST[1]

The PCM_OP_NUM_POL_DEVICE_ASSOCIATE policy opcode returns an error in the following cases:

- The number is already associated with the same type of service. For example, if the number is already associated with a fax service, an error is returned if you try to associate the number with another fax service.
- The service that is being associated does not belong to the same account as an existing associated service.
- The number network element is not the same as the network element of a SIM card that is already associated with the service.

Customizing Telephone Number Attributes

To customize telephone number attributes when a number is created, use the PCM_OP_NUM_POL_DEVICE_CREATE opcode. This policy opcode validates a new number to make sure it is unique in the database.

This policy opcode is called by the PCM_OP_DEVICE_CREATE opcode and calls the PCM_OP_NUM_POL_CANONICALIZE policy opcode.

You can customize this policy opcode if you extend the number device attributes and require additional validation or if you want to change existing validations.

The PCM_OP_NUM_POL_DEVICE_CREATE policy opcode returns an error if a new number is not unique in the database.

Customizing How a Number Can Be Changed

To customize how a number can be changed, for example, which digits can be changed, use the PCM_OP_NUM_POL_DEVICE_SET_ATTR policy opcode.

By default, the PCM_OP_NUM_POL_DEVICE_SET_ATTR opcode supports changing US area codes by using the following logic: If the number starts with 001 and is 13 digits long, allow changing digits 4 through 6.

This opcode is called by the PCM_OP_DEVICE_SET_ATTR opcode and calls the PCM_OP_NUM_POL_CANONICALIZE policy opcode.

You can customize the PCM_OP_NUM_POL_DEVICE_SET_ATTR policy opcode to allow customized modifications on numbers. For example, you can customize this

policy opcode to allow changing digits 5 through 7 of a number if the number starts with 44.

Changing a Block's Brand

To change a block's brand, and the brand of all numbers in the block, use the PCM_OP_NUM_POL_DEVICE_SET_BRAND policy opcode.

All numbers must be in a New or Unassigned state. This policy opcode is called by the PCM_OP_NUM_MODIFY_BLOCK opcode.

You can customize this policy opcode to change how numbers are associated with brands.

34

Installing and Configuring Number Manager and Number Administration Center

This chapter describes how to install and configure Oracle Communications Billing and Revenue Management (BRM) Number Manager and Number Administration Center.

For information about managing number inventory, see "About Managing Telephone Numbers".

Mandatory Configuration Tasks

In addition to installing the Number Manager software, you need to configure data before you can use Number Manager and Number Administration Center.

Note: You can perform these tasks in any order, but you must install Number Manager first.

- You must define and load network elements. (If you already installed SIM Manager, you might have already done this.) See "Creating Network Elements".
- You must set the default number quarantine period; for example, 90 days. See "Changing the Default Quarantine Period".
- You must define number device states. This requires two procedures: defining and loading the state transition model, and loading the localized state descriptions. See "Customizing Device States".
- You must define number device and service associations. See "Customizing Number Device Service Associations".
- You must define number categories. See "Creating Number Categories".
- You must define vanity number types. Even if you do not use vanity numbers, you must load the **num_vanities.***locale* file. See "Creating Vanity Types".

System Requirements

Number Manager is supported on the HP-UX IA64, Solaris, AIX, and Linux operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Number Administration Center is supported on Windows systems and requires 100 MB of disk space to download, extract, and install the software.

Note: Number Administration Center requires the Java Runtime Environment (JRE). It is included in the Number Administration Center package and is approximately 50 MB. If the JRE was already installed with another BRM client application, it will not be reinstalled.

Software Requirements

Before installing Number Manager, you must install:

- Third-Party software, which includes the PERL 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 or Oracle 11g.

Before installing Number Administration Center, you must install:

- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- (Windows only) An application such as WinZip for extracting compressed files.

Installing Number Manager on UNIX

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 Number Manager on UNIX:

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 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.

   Caution: 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 the temp_dir directory and enter this command:

   7.5.0_NumberMgr_*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 Number Manager is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or Number Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the Number 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 Number Manager installation is now complete.

Important: You should verify that the Number Manager FMs were added to the Connection Manager (CM) configuration file.

Installing Number Administration Center

To install Number Administration Center on Windows:

1. Download the software to a temporary directory *(temp_dir)*.
2. Extract the downloaded **.zip** file to a temporary directory (*temp_dir*).
3. Go to *temp_dir* and run the **setup.exe** program. The installation wizard for Number Administration Center starts.

4. Answer the prompts in the installation wizard screens.

Your Number Administration Center installation is now complete.

Configuring Event Notification for Number Manager

Important: This is a mandatory configuration task.

Note: If you already configured event notification for SIM Manager, skip this procedure.

To use Number Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them. See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Ensure that the merged file includes the entire event notification list in the *BRM_Home*/**sys/data/config/pin_notify** file.
3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list. See "Editing the Event Notification List" in *BRM Developer's Guide*.
4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger. See "Triggering Custom Operations" in *BRM Developer's Guide*.
5. Load your final event notification list into the BRM database. See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Customizing Number Quarantine

You can customize how numbers are quarantined in the following ways:

- You can change the default quarantine period. See "Changing the Default Quarantine Period".
- You can manually unquarantine numbers. "Changing the Quarantine Status Manually".
- You can customize policy source code to change how number quarantine works. "Changing How Number Quarantine Works".

For more information, see "About Managing Telephone Numbers".

Changing the Default Quarantine Period

Important: This is a mandatory configuration task.

By default, the quarantine period is 90 days. You can change the default quarantine period. You can use a different default quarantine period for each brand.

To customize the default quarantine period, you run the **load_pin_num_config** utility to load the contents of the **pin_num_config** file into a **/config/num** object in the BRM database.

> **Important:** The utility needs a configuration file in the directory from which you run the utility. The login that you enter in the configuration file specifies the brand that the quarantine period applies to. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide* and "Running Utilities with a Branded Database" in *BRM Managing Customers*.

1. Edit the **pin_num_config** file in *BRM_Home*/**sys/data/config**. The default entry is:

   ```
   quarantine_period 90
   ```

   To change the quarantine period, change the number. For example:

   ```
   quarantine_period 45
   ```

2. Save the file.
3. Use the following command to load the **pin_num_config** file:

   ```
   load_pin_num_config pin_num_config
   ```

   For more information, see "load_pin_num_config".

To verify that the new value was loaded, you can display the **/config/num** 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*.) The **/config/num** object shows the quarantine period in seconds in the PIN_FLD_VALUE field. In this example, 7776000 seconds = 90 days.

```
0  PIN_FLD_VALUE    STR  [0]  "7776000"
```

Changing the Quarantine Status Manually

If you run out of new or unassigned numbers, use the **pin_change_num_quarantine** utility to make numbers available for unquarantine. When the numbers are available for unquarantine, you run the **pin_deferred_act** utility to change the status to unquarantined. Therefore, you first run the **pin_change_num_quarantine** utility, then you run the **pin_deferred_act** utility.

> **Note:** Both utilities need 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*. The login that you enter in the configuration file specifies the brand that the quarantine change applies to. See "Running Utilities with a Branded Database" in *BRM Managing Customers*.

When you use the **pin_change_num_quarantine** utility, you have the following options:

- You can use the **-l** parameter to display how many numbers will be unquarantined without committing anything to the database.

- When you make numbers available for unquarantine, remember that there might already be numbers that are available for unquarantine, and they will be unquarantined when you run the **pin_deferred_act** utility. You can use the **-a** parameter to take into account all numbers, even those that are ready for unquarantine.

  For example, you might want to unquarantine 50 numbers, but you have 30 numbers ready for unquarantine already.

  - If you use the **-a** parameter, the utility affects only 20 numbers because 30 are ready to be unquarantined already. Your total quantity is 50.
  - If you do not use the **-a** parameter, the utility unquarantines 50 numbers in addition to the 30 that are already available for unquarantine. Your total quantity is 80.

For more information, see "Examples".

1. Use the following command to run the **pin_change_num_quarantine** utility:

   ```
   pin_change_num_quarantine Quantity_of_numbers
   ```

   For example, to unquarantine 100 numbers, enter:

   ```
   pin_change_num_quarantine 100
   ```

   For more information, see "pin_change_num_quarantine".

   Note: If there are fewer numbers available, the utility displays a warning.

2. Use the following command to run the **pin_deferred_act** utility:

   ```
   pin_deferred_act
   ```

   There are no input parameters for this utility. For more information, see "pin_deferred_act" in *BRM Configuring and Running Billing*.

Changing How Number Quarantine Works

By default, you can assign a quarantined number to an account. You can modify the PCM_OP_NUM_POL_DEVICE_ASSOCIATE policy opcode to not allow assigning a quarantined number to an account.

In addition, you can create your own opcode to handle number quarantine. By default, the PCM_OP_NUM_QUARANTINE opcode controls how numbers are quarantined. To customize how number quarantine works, you can edit and load the **pin_device_state_num** file to call a custom opcode instead of calling PCM_OP_NUM_QUARANTINE. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

Customizing Device States

You can customize number device states to support custom business logic. For example, you can change how numbers are quarantined by adding custom device states.

You can create brand-specific device states. See "Device Management and Brands" in *BRM Developer's Guide*. However, you need to be careful to create device state mappings that work if you change the brand of a number. For example, if a device is in

state 2 in one brand, it is still in that state when you change brands, but state 2 might be defined differently.

To customize device states, you edit the **pin_device_state_num** file in *BRM_Home*/**sys/data/config** and load it by using the **load_pin_device_state** utility. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

If a Number Has No Status Displayed in Number Administration Center

If a number has no status displayed in Number Administration Center, it means the device states for the number's brand has not been loaded. The brand administrator for the brand in question must do the following:

1. Load the **pin_device_state_num** file for that brand.
2. Load the **num_device_states.***locale* file if it changed or was not previously loaded.
3. Restart the Connection Manager (CM).
4. Restart the Number Administration Center.

Adding Device State Names

Important: This is a mandatory configuration task.

When you customize device states, you need to add the new device state to the list of device states displayed in Number Administration Center. To do so, you edit the **num_device_states.en_US** sample file in the *BRM_Home*/**sys/msgs/numdevicestates** directory. 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 num_device_states.locale
```

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

Note: 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*.

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

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

Creating Network Elements

Important: This is a mandatory configuration task.

Note: If you already created network elements for SIM cards, you do not need to create them again. The same network elements are used for numbers and SIM cards.

By default, there are only sample network elements. Before creating numbers, you need to add the network elements that you will use with your GSM services.

Note: You cannot define network elements by brand; all network elements can be used by any brand.

To customize network elements, you edit the **pin_network_elements** file, then run the "load_pin_network_elements" utility to load the contents of the file into a **/config/network_element** object in the BRM database.

Note: The 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*.

1. Edit the **pin_network_elements** file in *BRM_Home*/**sys/data/config**. The **pin_network_elements** file includes examples and instructions.

   The file includes a list of network elements. For example:

   ```
   sample_network_element_1
   sample_network_element_2
   ```

   Caution: The **load_pin_network_elements** utility overwrites existing network elements. If you are updating network elements, you cannot load new network elements only. You must load complete sets of network elements each time you run the **load_pin_network_elements** utility.

2. Save the **pin_network_elements** file.
3. Use the following command to load the **pin_network_elements** file:

```
load_pin_network_elements pin_network_elements
```

To verify that the network elements were loaded, you can display the **/config/network_element** 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*.) This example shows a network element in the **/config/network_element** object:

```
PIN_FLD_NETWORK_ELEMENT   STR  [0]  "Sample_Network_Element"
```

Creating Number Categories

Important: This is a mandatory configuration task.

You can customize the list of number categories displayed in Number Administration Center. For information about number categories, see "Creating a Block of Telephone Numbers".

The default number category is **None**. The only other default option is **Reserved**. To customize number categories, you edit the **num_categories.en_US** sample file in the *BRM_Home*/**sys/msgs/numcategories** directory. 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*.

> **Note:** You cannot create brand-specific number categories. Number categories displayed in Number Administration Center are available to all brands.

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

```
load_localized_strings num_categories.locale
```

> **Note:** 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*.

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

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

Creating Vanity Types

> **Important:** This is a mandatory configuration task.

You can customize the list of vanity types displayed in Number Administration Center. For information about vanity types, see "Creating a Block of Telephone Numbers".

To create vanity types, you edit the **num_vanities.en_US** sample file in the *BRM_Home*/**sys/msgs/numvanities** directory. 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*.

> **Note:**
>
> - You cannot create brand-specific vanity types. Vanity types displayed in Number Administration Center are available to all brands.
> - You can customize the PCM_OP_NUM_POL_DEVICE_CREATE policy opcode to apply vanity types automatically. For example, you could automatically apply a vanity type to numbers ending in 2000.

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

```
load_localized_strings num_vanities.locale
```

> **Note:** 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*.

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

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

Customizing Number Device Service Associations

> **Important:** This is a mandatory configuration task.

By default, you can associate numbers with the following services:

- **/service/telco/gsm/telephony**
- **/service/telco/gsm/sms**
- **/service/telco/gsm/fax**
- **/service/telco/gsm/data**

You can change which services can be associated with any number device in the BRM system. To do so, edit the **pin_device_permit_map_num** file and load it by using the **load_pin_device_permit_map** utility. See "Defining Device-to-Service Associations" in *BRM Developer's Guide*.

> **Note:** Device service associations are brand-aware and are associated with the root brand by default. See "Device Management and Brands" in *BRM Developer's Guide*.

Customizing How Service Types Are Used

By default, values entered in the **Service type** box do not have any business logic applied to them. You can customize the Number Manager business policies to use the value entered in the **Service type** box. For example, you might create service types for voice-only numbers or data-only numbers.

You can customize the following Number Manager policy opcodes to add business logic to service types:

- PCM_OP_NUM_POL_DEVICE_ASSOCIATE
- PCM_OP_NUM_POL_DEVICE_CREATE
- PCM_OP_NUM_POL_DEVICE_SET_ATTR

Changing the Number Administration Center Number Display Format

To change the Number Administration Center number display format, edit the Number Format entry in the **NumAdmin.properties** file. The Number Format entry specifies the default Java class that defines the Number Administration Center number display. To use a different number format, create a Java class and specify it in the Number Format entry.

The default mask is:

xxx-xxx-xxx-xxxx

How BRM Chooses a Mask

When you specify several masks, BRM chooses the mask that best matches the input number. For example, if the input number is:

044.112.233.3444

and the list of masks is:

(091)xx-xx-xxx-xxx

(044)xx.xx.xxx.xxx

(212.xxx.xxxx)

the mask used is:

(044)xx.xx.xxx.xxx

About Truncating Numbers by Using Masks

When setting the number format, you can truncate numbers. For example, if a number is entered as 001(408)5551212, you can truncate it to display as 5551212. Numbers are truncated by default.

When you truncate numbers, you can specify where to truncate: the left side of the number or the right side. By default, numbers are read from right to left, so the left side of the number is truncated.

Figure 34–1 shows the difference between truncating right to left and left to right. (A value of **true** truncates right to left.)

Figure 34–1 Truncating Numbers Differences

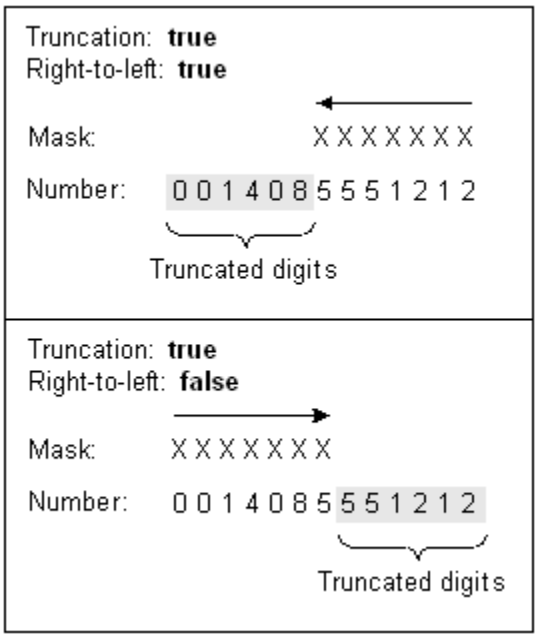


Table 34–1 shows examples of how numbers are formatted based on the mask and truncation. A truncate value of **true** means right-to-left.

Table 34–1 Number Truncation Examples

Input	Mask	Truncated	Truncate direction	Result
1(408)5179011	xxx.xxx.xxxx	true	true (<<<)	408.517.9011
1(408)5179011	xxx.xxx.xxxx	false	true (<<<)	1408.517.9011
234-5678	x-xxx-xxx-xxxx	false	true (<<<)	234-5678
234-5678	x-xxx-xxx-xxxx	false	false (>>>)	2-345-678
001(408)517901100	001-xxx-xxx-xxxx	true	false (>>>)	001-408-517-9011
001(408)517901100	001-xxx-xxx-xxxx	false	false (>>>)	001-408-517-901100
1(408)517-901100	x.xxx.xxx.xxxx	true	true (<<<)	0.851.790.1100
1(408)517-901100	x.xxx.xxx.xxxx	false	true (<<<)	140.851.790.1100

Changing the Number Display Format

To change the number display format:

1. Open the **NumAdmin.properties** file in the **C:\Program Files\Common Files\Portal Software** directory.
2. Edit the **mask** entries as needed. The default mask looks like this:

   ```
   device.num.formatter.mask.default=xxx-xxx-xxx-xxxx
   ```

3. (Optional) To add a new mask, add the mask name to the **device.num.formatter.masks** entry.

   This example shows a new mask for Denmark:

   ```
   device.num.formatter.masks=default france uk denmark
   ```

This example shows the mask definition:

```
device.num.formatter.mask.denmark=45-xxxx-xxxxx
```

4. (Optional) If a number is larger than the mask, you can specify whether to truncate the number and how to truncate it:

   - To truncate numbers that are longer than the mask, edit the **device.num.formatter.truncate** entry. Enter **true** to truncate numbers, enter **false** to not truncate numbers.

     By default, numbers are not truncated:

     ```
     device.num.formatter.truncate=false
     ```

   - To specify how to truncate the number, edit the **device.num.formatter.direction** entry. By default, numbers are read from right to left, and truncation occurs at the left side of the number.

     For example, if the number is 0014085551212 and the mask is xxx-xxxx, the truncated number is 5551212.

     Enter **true** to truncate right-to-left, enter **false** to truncate left-to-right:

     ```
     device.num.formatter.direction=true
     ```

5. Save and close the file.
6. Restart Number Administration Center.

Creating a Custom Number Format Class

You can create and use a custom number format Java class. You might want to do this if defining custom masks is not sufficient.

1. Create the class.
2. Open the **NumAdmin.properties** file in the **C:\Program Files\Common Files\Portal Software** directory.
3. Edit the **device.num.formatter.class** entry:

   ```
   device.num.formatter.class=custom_Class
   ```

   where *custom_Class* is your class.
4. Edit the **device.num.formatter.method** entry to use the method defined in your class. See "Changing the Number Display Format".

   ```
   device.num.formatter.method=format
   ```

5. Edit the **device.num.formatter.masks** entry and add new masks, as described in "Changing the Number Administration Center Number Display Format".
6. Save and close the file.
7. Restart Number Administration Center.

Configuring Search Performance in Number Administration Center

You can edit the **NumAdmin.properties** file to change how Number Administration Center searches for blocks and numbers.

1. Open the **NumAdmin.properties** file in the **C:\Program Files\Common Files\Portal Software** directory.
2. To set the default quantity of numbers displayed in a page of search results, edit the **pageSize** entry. For example:

   ```
   pageSize=50
   ```

   You can change the page size at any time when using Number Administration Center.
3. To change the search performance, edit the **searchSize** entry. For example:

   ```
   searchSize=100
   ```

   If you use a low-bandwidth system with a lot of numbers, reducing the step size can speed up searches.
4. Save and close the file.
5. Restart Number Administration Center.

Uninstalling Number Manager

To uninstall Number Manager on UNIX, run the *BRM_Home*/**uninstaller/NumberMgr/uninstaller.bin**.

35

Number Manager Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Number Manager utilities.

load_pin_num_config

Use this Oracle Communications Billing and Revenue Management (BRM) utility to change the default telephone number quarantine period. You can use different default quarantine periods for each brand.

For more information, see:

- Changing the Default Quarantine Period
- About Number Device States

> **Note:** To connect to the BRM database, the **load_pin_num_config** 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*. The login that you enter in the configuration file specifies the brand that the quarantine period applies to. See "Running Utilities with a Branded Database" in *BRM Managing Customers*.

Location

BRM_Home/**bin**

Syntax

```
load_pin_num_config [-d] [-v] filename
```

Parameters

-d

Enables debugging mode.

-v

Displays information about successful or failed processing as the utility runs.

> **Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename***.log** with the name of the log file:
>
> **pin_change_num_quarantine** other_parameter **–v >** *filename***.log**

filename

The file (**pin_num_config)** that includes the value to use for the default quarantine period.

Results

Reports success or displays an error.

pin_change_num_quarantine

Use this Oracle Communications Billing and Revenue Management (BRM) utility to make telephone numbers available for unquarantine before the default quarantine period is over.

For more information, see:

- Changing the Quarantine Status Manually
- About Number Device States

> **Note:** To connect to the BRM database, the **pin_change_num_quarantine** 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

```
pin_change_num_quarantine [-d] [v] [l] [a] Quantity_of_numbers
```

Parameters

-d

Enables debugging mode.

-v

Displays information about successful or failed processing as the utility runs.

> **Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename*.**log** with the name of the log file:
>
> **pin_change_num_quarantine** other_parameter **–v >** *filename*.**log**

-l

Displays the results, but does not change the quarantine.

-a

Includes all numbers, even those that are ready for unquarantine. For example, you might want to unquarantine 50 numbers, but you have 30 numbers ready for unquarantine already.

If you use the **-a** parameter, the utility affects only 20 numbers, because 30 are ready to be unquarantined already. You total quantity is 50.

If you *do not* use the **-a** parameter, the utility unquarantines 50 numbers in addition to the 30 that are already available for unquarantine. Your total quantity is 80.

For more information, see "Examples".

Quantity_of_numbers

The quantity of numbers that you want to make available for unquarantine. You can also use this number in combination with the **-l** parameter to find out how many numbers are available for quarantine.

> **Tip:** To find out how many numbers are available for unquarantine, enter a very large number.

Results

The utility makes the specified quantity of numbers available, or reports how many numbers are available.

If there are fewer numbers available than specified, this utility displays a warning.

Examples

- **Listing the available quantity of numbers that can be unquarantined**

  Command:

  ```
  pin_change_num_quarantine -l 100
  ```

  Results:

  ```
  100 number(s) available for premature unquarantining
  ```

- Listing the available quantity of numbers that can be unquarantined, including numbers that are already available for unquarantine

  Command:

  ```
  pin_change_num_quarantine -la 100
  ```

Results:

```
40 number(s) found ready to be unquarantined
60 number(s) available for premature unquarantining
```

- Make 100 numbers available for unquarantine, in addition to any numbers already available for unquarantine

  Command:

  pin_change_num_quarantine 100

  Results:

  ```
  100 number(s) updated successfully
  ```

- Make 100 numbers available for unquarantine, taking into account numbers that are already available

  Command:

  pin_change_num_quarantine -a 100

  Results:

  ```
  40 number(s) found ready to be unquarantined
  60 number(s) updated successfully
  ```

Part VIII

Managing SIM Card Inventory

Part VIII describes how to manage SIM card inventory in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

36

About Managing SIM Card Inventory

This chapter describes how to manage SIM card inventory in your Oracle Communications Billing and Revenue Management (BRM) system by using SIM Manager and SIM Administration Center.

Important: SIM Manager and SIM Administration Center are optional components, not part of base BRM.

You need to manage SIM card inventory when you offer GSM services. For information about GSM Manager, see "About Integrating Wireless Services".

Before using SIM Manager and SIM Administration Center, you need to know the following information:

- Basic BRM concepts. See *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- How to create and edit BRM configuration files. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

About Managing SIM Cards

To manage SIM cards, you use the following components:

- SIM Manager includes the opcodes and storable classes required for creating and managing SIM card devices in the BRM database.
- SIM Administration Center is a GUI application that you use to create and process SIM card orders, and to change SIM card brand and network element.

You use Customer Center to assign and manage SIM cards in customer accounts. For more information, see information about working with GSM accounts in the Customer Center Help.

About Creating SIM Cards

Use SIM Administration Center to order SIM cards from a vendor. For complete instructions on using SIM Administration Center, see the SIM Administration Center Help.

To order SIM cards:

1. Create an order in SIM Administration Center.

   The order is stored in the BRM database and can be updated or canceled.

2. Create one or more *vendor request files*. This file specifies the quantity of SIM cards, the starting SIM card and IMSI numbers, and the SIM card attributes, such as the carrier name and SIM card format. This text file follows the standard GEMPLUS file format for SIM card orders.

   Vendor request files cannot specify more than 5000 SIM cards. If you create an order for more than 5,000 cards, the order is automatically divided into multiple files. For example, if you create an order for 50,000 SIM cards, SIM Administration Center creates 10 vendor request files of 5,000 SIM cards each. (See "About SIM Card Request and Response Files".)

   You can create a maximum order of 499,995,000 SIM cards, which would create 99,999 vendor request files.

   You can change the default values for maximum quantity of SIM cards in an order and maximum number of SIM cards in a request file. See "Customizing SIM Administration Center".

3. Send the vendor request files to the vendor. You can use any file transfer method, such as FTP or email.

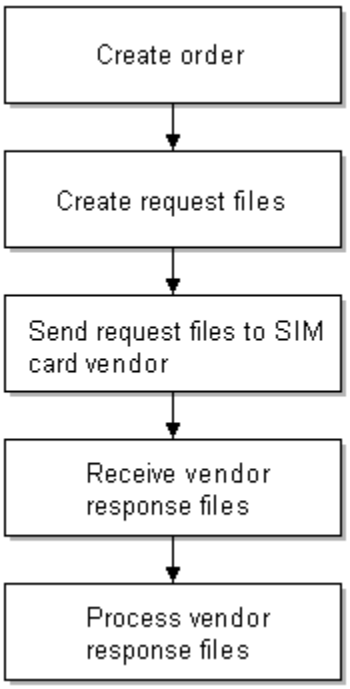
4. Receive the *vendor response* files and process them in SIM Administration Center. This creates a **/device/sim** object for each SIM card.

   Note: Multiple vendor *request* files are created automatically, based on the size of the order. However, you need to process each vendor *response* file individually.

   You can pre-provision SIM cards when you process the vendor response files. When you pre-provision a SIM card, the SIM card is assigned a SIM card number and an IMSI number on the network. For more information, see "About SIM Card Pre-Provisioning".

Figure 36–1 shows the process for creating SIM cards:

Figure 36–1 SIM Card Creation Process



About SIM Card Orders

> **Note:** Contact your SIM card vendor for information about the SIM card attributes. For example, you need to define your graphic elements (the logo and graphics that are printed on the card).

When you create an order, you can specify the following order attributes:

- Customer name and address information. This can be your company, or a Branded service management site.

The name and address information for the vendor who creates the SIM cards.

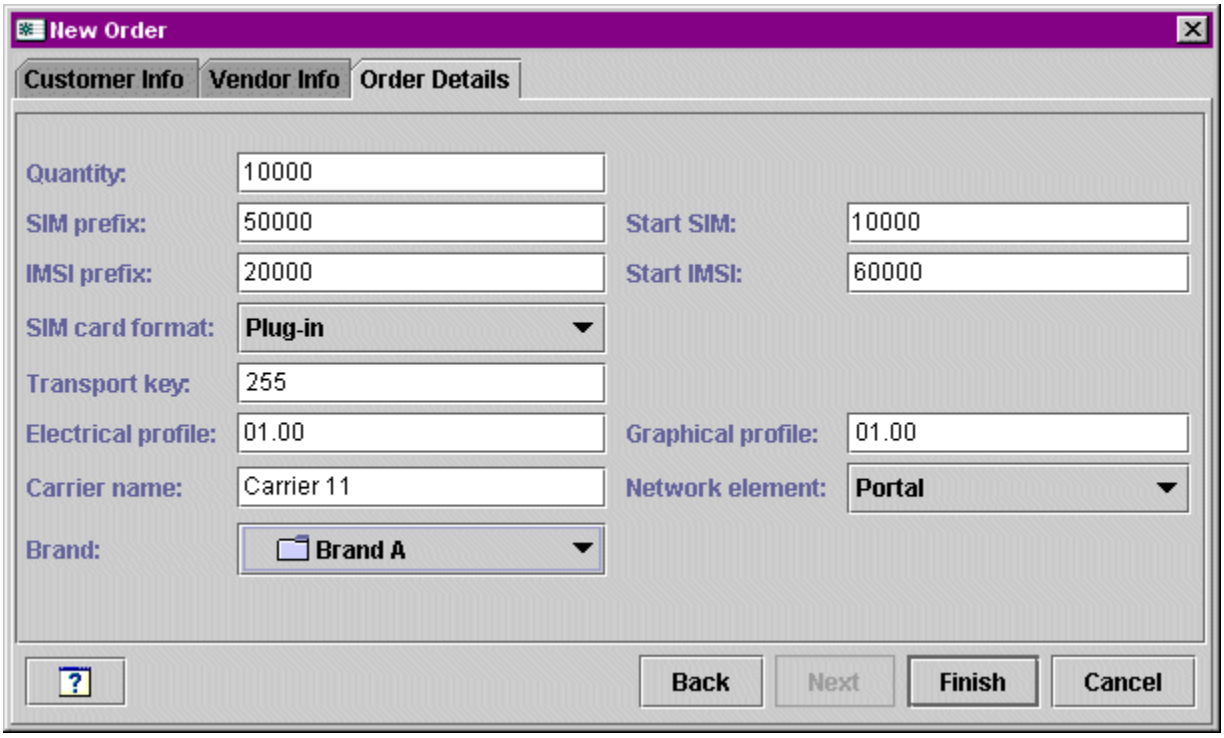
> **Note:** There are limitations on how many characters you can use for each piece of customer or vendor information. However, these limitations allow for more characters than you will probably use. For more information, see the **/order** object definition.

- The quantity of SIM cards in the order. By default, the maximum is 499,995,000 SIM cards.
- Starting SIM card number and IMSI.
- Brand.
- SIM card attributes:
  - SIM card format, for example, ISO or Plug-in
  - Carrier name
  - Transport key

- Electrical profile
- Graphical reference, for example, a company logo
- SIM card number prefix
- IMSI prefix
- Network element

Figure 36–2 shows the **Order Details** tab used for specifying SIM card attributes in SIM Administration Center:

Figure 36–2 Order Details Tab



About Managing Orders

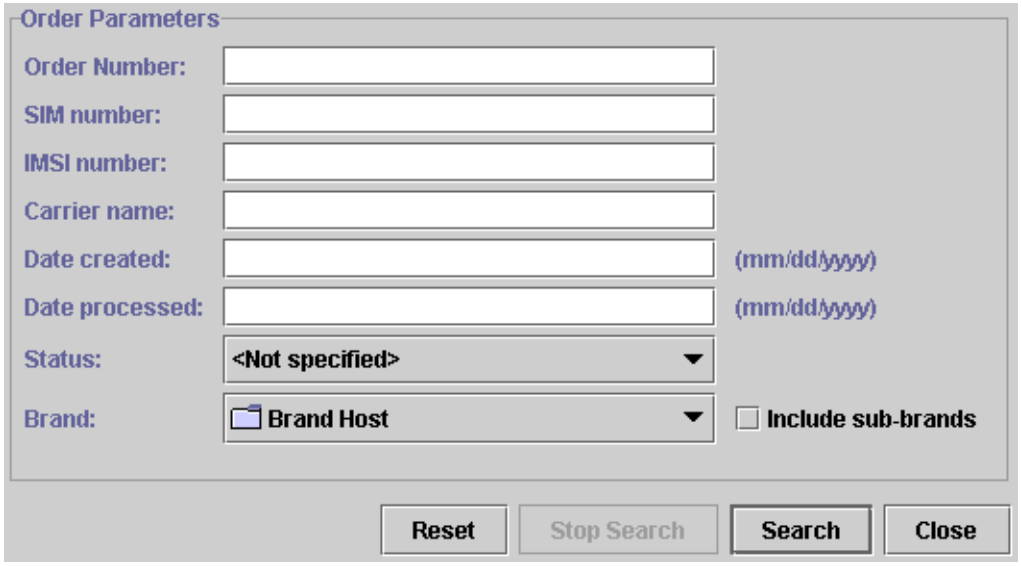
When you create and manage SIM card orders, BRM assigns a status to the order. SIM Administration Center displays the order status.

1. When you create an order, the status is *New*. You can save a new order, or cancel it. When you cancel an order, the order status is set to *Canceled*. You can also modify an order if it is in the New status. You cannot delete an order.
2. After you create a request file, the order status is set to *Requested*. When an order has the Requested status, you cannot cancel or modify it.
3. When you begin processing vendor response files, the order status is set to *Partially Received*.
4. After you process all the vendor response files in an order, the order status is set to *Received*.

   When the order has a Received or Partially Received status, you cannot change the order attributes. However, after you process the vendor response file, you can change the SIM card brand and network element.

You can search for orders of any status. Figure 36–3 shows the attributes you use when searching for an order:

Figure 36–3 Order Search Attributes



Note: SIM card orders are stored in the database as **/order/sim** objects.

About SIM Card Request and Response Files

You manage orders with your SIM card vendor by exchanging request and response files. The request files include information about the order, and are used by the vendor to create the SIM cards. The response files contain information from the vendor about the SIM cards.

SIM Administration Center automatically generates request files using the following naming convention:

*Customer_abbreviationXXXXX***.inp**

- The customer abbreviation is the first three letters from the name in the **Customer name** field when creating the order.
- The customer abbreviation is followed by five digits, incremented for each file in the order. The first file in an order uses 00001.

For example, if the customer name is "Oracle," the first file in an order is named **Ora00001.inp**.

Important: All orders that you create for the same customer use the same file names. Therefore, when creating request files, you should always put them in a new or empty folder for each order, so files from previous orders are not overwritten.

The vendor response files use the following naming convention:

*Customer_abbreviationXXXXX***.out**

- The customer abbreviation is the first three letters from the name in the **Customer name** field when creating the order.
- The customer abbreviation is followed by five digits, incremented for each order. The first file in an order uses 00001.

For example, if the customer name is “Oracle,” the first file in an order is named **Ora00001.out**.

About SIM Card Pre-Provisioning

When you process vendor response files, you can also pre-provision SIM cards. This allows a customer to perform tasks such as calling a service center with their new phone.

Note: You can create brand-specific pre-provisioning services. In that case, all SIM cards for a brand are associated with the same service.

Important:

- If you pre-provision SIM cards, you must define the pre-provisioning service. See "Specifying the Pre-Provisioning Service".
- If you do not pre-provision SIM cards, you must change the device state transition. See "Customizing How to Pre-Provision SIM Cards".

When a SIM card is pre-provisioned, the following actions occur:

1. BRM sends information about each SIM card to the Provisioning Data Manager (**dm_prov_telco**). This information includes the IMSI, SIM card number, an encrypted KI value, the network element, and a dummy MSISDN number.

   Note: The dummy MSISDN number is a placeholder. You can change the MSISDN number if you customize pre-provisioning. See "Customizing How to Pre-Provision SIM Cards".

2. The Wireless Provisioning DM sends the provisioning request to the provisioning system, which communicates with the network to provision the SIM card.
3. BRM updates the SIM card device state to one of the following:
   - If provisioning succeeds, the SIM card device state is set to *Released*. A CSR can assign this SIM card to a customer. Because the customer does not need to know about the pre-provisioning service, the SIM card device in the BRM database is disassociated from the pre-provisioning service object.
   - If provisioning fails, the SIM card device state is set to *Failed Provisioning*.

Enabling SIM Card Pre-Provisioning

To enable SIM card pre-provisioning:

1. Configure PCM_OP_SIM_DEVICE_PROVISION (opcode number 2403) in state transition **0** (Raw) -> **1** (New) by editing the **pin_device_state_sim** file and loading it with the **load_pin_device_state** utility.

   This example shows how to enable pre-provisioning:

   ```
   New -> Assigned or Provisioning
   1: 1: 1: 1: 2403: 0
   2: 0: 0
   5: 0: 0
   3: 0: 0
   ```

2. Configure the following information in the *BRM_Home*/**sys/data/config/pin_notify** file (if not already configured) for state transition and load it with **load_pin_notify**.

   ```
   2706    0       /event/notification/device/state
   ```

3. In the *BRM_Home*/**sys/data/config/pin_telco_provisioning** configuration file, specify the pre-provisioning information for each device type using the following syntax. A separate line is required for each field, even for the same device type.

   ```
   Device provisioning info:
   DeviceType, DeviceState, ProvAction, Field1, "String1",
   DeviceType, DeviceState, ProvAction, Field2, "String2"
   ```

   where:

   - *DeviceType* specifies the type of device to be pre-provisioned.
   - *DeviceState* specifies the device state that triggers the generation of a service order. If you do not specify the **DeviceState** value, service orders are generated for all device state transitions.
   - *ProvAction* specifies the provisioning action for the device objects in the service order. In this case, you always use the value **P**, for provisioning.
   - *FieldX* specifies the name of a field that contains a device attribute to include in the service order. Field names are replaced by actual values when you run **pin_telco_provisioning**.
   - *StringX* specifies a string that is used in the service order to identify the attribute specified by the corresponding Field value. You can query for the string in the service order.

   For example, the following lines add the authentication key, IMSI, and device ID values to the service order when a SIM card is pre-provisioned.

   ```
   /device/sim, 3, P,PIN_FLD_DEVICE_SIM.PIN_FLD_KI "KI",
   /device/sim, 3,P,PIN_FLD_DEVICE_SIM.PIN_FLD_IMSI "IMSI",
   /device/sim, 3,P,PIN_FLD_DEVICE_SIM.PIN_FLD_DEVICE_ID "SIM"
   ```

4. Load the file into the provisioning configuration object by using the **load_pin_telco_provisioning** utility.

About Managing SIM Cards in a Branded System

When you log in to SIM Administration Center, your login associates you with a brand. When you create and process an order, you can assign the order to the login brand, or any sub-brand. All SIM cards in the order are also associated with the same brand.

You cannot change the brand of an order.

You can change the brand of SIM card devices if the device state is Released. After a SIM card has been assigned to a customer, you cannot change the brand, even if the device state is Unassigned. See "Changing the Brand or Network Element of SIM Cards".

You cannot create brand-specific sets of network elements or SIM card formats. All network elements and SIM card formats can be used by all brands.

See "About Branding" in *BRM Managing Customers*.

Managing SIM Cards in a Multischema System

If you manage SIM cards in a multischema system:

- You cannot change the brand of a SIM card to a brand that is hosted in a different database schema from the schema that the current brand is hosted in. For example, if a SIM card belonging to Brand A is in schema 0.0.0.1, you cannot change the SIM card to Brand B if Brand B is in schema 0.0.0.2.
- If you log in as a brand host administrator and select Brand Host as the brand to create an order, you can create orders only in the schema that you logged in to. If you select any other brand, the order is created in the schema that hosts the selected brand.
- When searching for SIM cards or orders, if you log in as a brand host administrator and you select the top-level brand without including sub-brands in the search criteria, the search results include only SIM cards or orders from the schema you logged in to.
- In a multischema environment that is not brand enabled, you can create SIM cards only in the schema you are logged in to.

About Associating a SIM Card with a Customer's Service

When an account is created with a GSM service, or a SIM card is added to an existing account, the SIM card is provisioned, and associated with a service. The SIM card device state is set to *Assigned*.

When you associate a SIM card with a service, you usually also associate a number. See "About Associating SIM Cards and Numbers with Services".

About SIM Card Device States

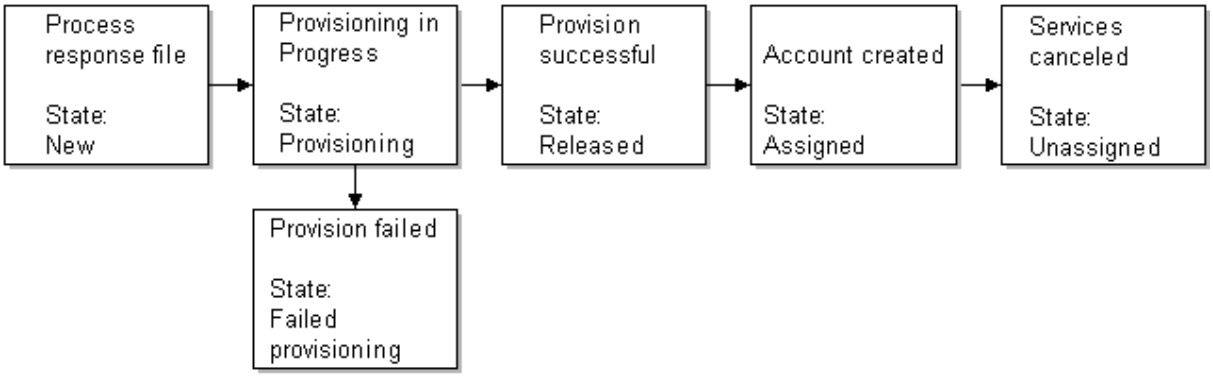
When you manage SIM cards, the SIM card devices have the following device states:

- When you process a vendor response file, you create SIM card devices in the database. The device state of the SIM cards is *New*.
- While a SIM card is being provisioned, the device state of the SIM cards in the file is set to *Provisioning*. After provisioning, a card is in one of two states:
  - If provisioning succeeds, the device state is set to *Released*.
  - If provisioning fails, the device state is set to *Failed Provisioning*. The device object is created in the database, but it cannot be assigned. (You can change the device state transitions to enable failed SIM cards to be assigned. See "Customizing SIM Card Device States".)
- When an account is created and assigned a SIM card, the device state is set to *Assigned*. If the account is inactive, the SIM card status remains Assigned.

- When all services associated with the SIM card are canceled, the SIM card is disassociated from all services and the device state is set to *Unassigned*. (If the SIM card is associated with multiple services, the SIM card remains Assigned as long as one service is active.) By default, you cannot assign a SIM card when it is in the Unassigned state. (You can change the device state transitions to allow unassigned SIM cards to be assigned. See "Customizing SIM Card Device States".)

Figure 36–4 shows the progression of SIM card device states:

Figure 36–4 SIM Card Device State Progression



Changing the Brand or Network Element of SIM Cards

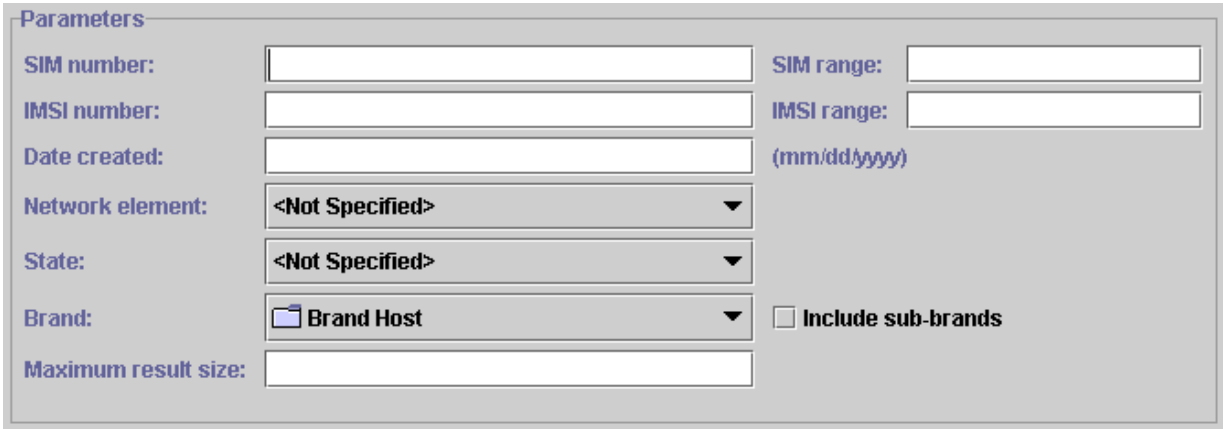
You use SIM Administrator to change the following SIM card attributes:

- The network element.
- The brand associated with the SIM cards.

> **Note:** You cannot use the SIM Administration Center to associate SIM cards with accounts and services, or change the status of a SIM card. To do those tasks, use Customer Center.

To change the SIM card brand or network element, you search for the SIM card. You can base your search on the SIM card attributes, as shown in Figure 36–5:

Figure 36–5 SIM Card Search Attributes



From the search results, select the SIM cards that you want to modify. You can only change the brand and network elements for SIM cards in the Released state.

For complete instructions on using SIM Administration Center, see the SIM Administration Center Help.

About SIM Card Customization Options

You can customize how you manage SIM Card inventory as follows:

- You can customize network elements.

  Important: By default, there are only sample network elements; you must add your own. See "Creating Network Elements".

- You can customize SIM card device and service associations. See "Customizing SIM Card and Service Associations".
- You can customize SIM card device states. See "Customizing SIM Card Device States".
- You can create custom SIM card formats. See "Creating SIM Card Formats".
- You can choose to not pre-provision SIM cards, or to pre-provisioning them with a different service, for example, a prepaid service. See "Customizing How to Pre-Provision SIM Cards".
- You can customize how SIM Administration Center works. See the following:
  - Specifying the Maximum Quantity of SIM Cards in a Request File
  - Specifying the Maximum Quantity of SIM Cards in an Order
  - Specifying the Information You Want to Receive about SIM Cards
  - Specifying whether the Response File Includes a Check Digit
  - Specifying the Size of Searches

Getting Information about SIM Card Usage

You can get information about SIM card inventory and status by running the SIM Manager reports. See "SIM Card Inventory Management Report" in *BRM Reports*.

In addition, you can display the history of a customer's SIM card in Customer Center. See information about displaying SIM card history in the Customer Center Help.

About the SIM Manager Opcodes

To create SIM cards in the BRM database, SIM Manager uses the following opcodes:

- PCM_OP_SIM_CREATE_ORDER. See "Creating and Updating SIM Card Orders".
- PCM_OP_SIM_UPDATE_ORDER. See "Creating and Updating SIM Card Orders".
- PCM_OP_SIM_PROCESS_ORDER_RESPONSE. See "Creating SIM Cards".
- PCM_OP_SIM_DEVICE_PROVISION. See "Provisioning SIM Cards".

Creating and Updating SIM Card Orders

When you use SIM Card Administrator to create an order, PCM_OP_SIM_CREATE_ORDER creates an order object (**/order/sim)** in the database. It returns an error if it finds a duplicate SIM card number or IMSI. The order status is set to New.

When you update an order, PCM_OP_SIM_UPDATE_ORDER updates the order object in the database. It returns an error if it finds a duplicate SIM card number or IMSI.

PCM_OP_SIM_UPDATE_ORDER also changes the order status, for example, when you process a response file, or cancel an order.

This opcode is called when a customer updates the order, or when the order status needs to be changed, for example, after processing a vendor response file. This opcode is also called when an order is canceled.

If no error is found, this opcode updates a SIM card order object. If the order is being canceled, this opcode changes the status to Canceled. This opcode will not function if the order is not in the New status.

Creating SIM Cards

SIM Card Manager uses PCM_OP_SIM_PROCESS_ORDER_RESPONSE to process the order response file and create SIM card device objects (**/device/sim** objects) in the database.

PCM_OP_SIM_PROCESS_ORDER_RESPONSE does the following:

1. Reads the status of the order from the order object. If an order has been canceled, or there is no matching order for the response file, this opcode terminates and reports an error.
2. Finds the initial state of the device as defined in the **/config/device_state/sim** object. If the state is New, the SIM card is pre-provisioned. To disable pre-provisioning, or customize how SIM cards are provisioned, change the initial state from New to a customized state. See "About SIM Card Pre-Provisioning".
3. Finds the network element associated with the order.
4. Calls PCM_OP_DEVICE_CREATE to create the SIM card devices (**/device/sim**) in the BRM database.
5. Calls PCM_OP_SIM_UPDATE_ORDER to change the status of the order to either Partially Received or Received.

Provisioning SIM Cards

SIM Card Manager uses PCM_OP_SIM_DEVICE_PROVISION to associate a SIM card with a service, and to disassociate the pre-provisioning service.

This opcode is called by PCM_OP_DEVICE_SET_STATE during the state transition from initial to new and from provisioning to release.

PCM_OP_SIM_DEVICE_PROVISION is used as the validation opcode for changing the device state from New to Provisioning and from Provisioning to Released or Failed Provisioning.

To customize pre-provisioning, you can write your own opcode and use your custom opcode as the validation opcode. See "Customizing How to Pre-Provision SIM Cards".

Customizing SIM Card Manager

Use the following opcodes to customize SIM Manager:

- PCM_OP_SIM_POL_DEVICE_ASSOCIATE. See "Customizing SIM Card Service Association".
- PCM_OP_SIM_POL_DEVICE_CREATE. See "Customizing SIM Card Number Changes".
- PCM_OP_SIM_POL_DEVICE_SET_ATTR. See "Customizing SIM Card Validation".
- PCM_OP_SIM_POL_DEVICE_SET_BRAND. See "Customizing SIM Card Brand Association".

Customizing SIM Card Service Association

Use the PCM_OP_SIM_POL_DEVICE_ASSOCIATE to change how SIM cards and services are associated.

- If the service is being associated, this opcode validates that the network element is the same as the network element for the number associated with the same service. This opcode then calls PCM_OP_DEVICE_SET_STATE to change the status from Released to Assigned.
- If the service is being unassociated, this opcode disassociates the service and calls PCM_OP_DEVICE_SET_STATE to change the status from Assigned to Unassigned.

To allow Pipeline Manager to find an account based on the IMSI or MSISDN in the CDR, this opcode copies the IMSI and MSISDN to the alias list array in the associated service object. The element ID is significant:

- The IMSI is copied to ALIAS_LIST[0]
- The MSISDN is copied to ALIAS_LIST[1]

PCM_OP_SIM_POL_DEVICE_ASSOCIATE returns an error in the following cases:

- It returns an error if the network element of the SIM card and telephone number are not the same.
- It returns an error if the service type is already associated with the SIM card. The only exception is when two **/service/telco/gsm/telephony** services are associated with the same SIM card.
- It returns an error when the device and the service to be associated with it are not owned by the same account.
- It returns an error when the device and the service to be associated with it are not associated with the same brand.

Customizing SIM Card Validation

Use the PCM_OP_SIM_POL_DEVICE_CREATE policy opcode to change validation rules for creating SIM card devices.

This opcode validates a device by validating the SIM card number, IMSI, KI, and network element values. For example, it makes sure that the numbers have the correct number of digits and are in the proper syntax. It also verifies that the SIM card does not already exist in the database.

This opcode is called by PCM_OP_DEVICE_CREATE when creating a SIM card device.

Note: For information about validating the network element when updating a device, see PCM_OP_SIM_POL_DEVICE_SET_ATTR.

Customizing SIM Card Number Changes

Use the PCM_OP_SIM_POL_DEVICE_SET_ATTR policy opcode to change how SIM cards and services are associated.

This opcode ensures that the SIM card number (PIN_FLD_DEVICE_ID) cannot be changed.

This opcode is called by PCM_OP_DEVICE_SET_ATTR when updating a SIM card device.

This opcode returns an error if an attempt is made to change SIM card number (PIN_FLD_DEVICE_ID).

Customizing SIM Card Brand Association

Use the PCM_OP_SIM_POL_DEVICE_SET_BRAND policy opcode to change how SIM cards can be associated with brands.

When changing the SIM card brand, this opcode validates that the SIM card device state is Released. See "About Managing SIM Cards in a Branded System".

This opcode returns an error if the device state is anything other than Released.

37

Installing and Configuring SIM Manager and SIM Administration Center

This chapter describes how to install and customize Oracle Communications Billing and Revenue Management (BRM) SIM Manager and SIM Administration Center.

For information about SIM Manager and SIM Administration Center, see "About Managing SIM Card Inventory".

Mandatory Configuration Tasks

In addition to installing the SIM Manager software, you need to configure data before you can use SIM Manager and SIM Administration Center.

Note: You can perform these tasks in any order, but you must install SIM Manager first.

- You must define and load network elements. (If you already installed Number Manager, you might have already done this.) See "Creating Network Elements".
- You must define SIM device states. This requires two procedures: defining and loading the state transition model, and loading the localized state descriptions. See "Customizing SIM Card Device States".
- You must define SIM device and service associations. See "Customizing SIM Card and Service Associations".
- You must define SIM card formats. See "Creating SIM Card Formats".
- If you pre-provision SIM cards, you must specify the pre-provisioning service. If you do not pre-provision SIM cards, you must change the SIM card device states. See "Customizing How to Pre-Provision SIM Cards".
- You must specify whether to pre-provision SIM cards, and if so, the pre-provisioning service. See "Customizing How to Pre-Provision SIM Cards".
- You must load the **order_sim_status.***locale* file. You use this file to localize order status descriptions in SIM Administration Center. Even if you do not run a localized version of SIM Administration Center, you need to load this file. See "Loading Order Status Definitions".
- If you use an Oracle database, you must allow for encrypted fields. (By default, the KI value is encrypted.) See "Supporting Encrypted KI Values with Oracle".

System Requirements

SIM Manager is supported on the HP-UX IA64, Linux, Solaris, and AIX operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

SIM Administration Center is supported on the Windows platform and requires approximately 100 MB of disk space to download, extract, and install the software.

Note: SIM Administration Center requires the Java Runtime Environment (JRE). It is included in the SIM Administration Center package and is approximately 50 MB. If the JRE was already installed with another BRM client application, it will not be reinstalled.

Software Requirements

Before installing SIM Manager, you must install:

- Third-Party software, which includes the PERL 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 or Oracle 11g.

Before installing SIM Administration Center, you must install:

- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- (Windows only) An application such as WinZip for extracting compressed files.

To implement SIM card provisioning, you need to install GSM Manager, including the wireless provisioning opcodes. See "Installing and Configuring GSM Manager and Provisioning Data Manager".

Installing SIM Manager on UNIX

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 SIM Manager on UNIX:

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 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.

   Caution: 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 the temp_dir directory and enter this command:

   7.5.0_SIMMgr_*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 SIM Manager is **opt/portal/7.5**.

   Note: The installation program does not prompt you for the installation directory if BRM or SIM Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the SIM 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 SIM Manager installation is now complete.

Installing SIM Administration Center

To install SIM Administration Center on Windows:

1. Download the SIM Administration Center software from the BRM Software Web site to a temporary directory (*temp_dir*).
2. Extract the downloaded **.zip** file to a temporary directory (*temp_dir*).
3. Go to *temp_dir* and run the **setup.exe** program. The installation wizard for SIM Administration Center starts.
4. Answer the prompts in the installation wizard screens.

Your SIM Administration Center installation is now complete.

Configuring Event Notification for SIM Manager

Important: This is a mandatory configuration task.

Note: If you already configured event notification for Number Manager, skip this procedure.

To enable SIM card provisioning, BRM uses event notification. Before you can use SIM Manager, you must configure the event notification feature as follows:

1. If your system has multiple configuration files for event notification, merge them. See "Merging Event Notification Lists" in *BRM Developer's Guide*.
2. Ensure that the merged file includes the entire event notification list in the *BRM_Home*/**sys/data/config/pin_notify** file.
3. (Optional) If necessary to accommodate your business needs, add, modify, or delete entries in your final event notification list. See "Editing the Event Notification List" in *BRM Developer's Guide*.
4. (Optional) If necessary to accommodate your business needs, create custom code for event notification to trigger. See "Triggering Custom Operations" in *BRM Developer's Guide*.
5. Load your final event notification list into the BRM database. See "Loading the Event Notification List" in *BRM Developer's Guide*.

For more information, see "Using Event Notification" in *BRM Developer's Guide*.

Customizing SIM Card and Service Associations

Important: This is a mandatory configuration task.

By default, SIM card devices can be associated with the following services:

- **/service/telco/gsm/telephony**
- **/service/telco/gsm/fax**
- **/service/telco/gsm/data**
- **/service/telco/gsm/sms**

To change the services that a SIM card device can be associated with, edit the **pin_device_permit_map** file and load it by using the **load_pin_device_permit_map** utility. See "Defining Device-to-Service Associations" in *BRM Developer's Guide*.

Supporting Encrypted KI Values with Oracle

Important: This is a mandatory configuration task.

By default, the field in the **/device/sim** object that stores the KI value (PIN_FLD_KI) is encrypted. If you use an Oracle database, you must edit the Oracle DM configuration file to allow for encrypted fields. Otherwise, you cannot create SIM cards.

Note: Alternatively, you can unencrypt the PIN_FLD_KI field, but this is not recommended for security reasons. See "Encrypting Fields" in *BRM Developer's Guide*.

1. Open the Oracle DM configuration file (*BRM_Home*/**sys/dm_oracle/pin.conf**).
2. Uncomment the appropriate line for your operating system:

   ```
   #- crypt md5| libpin_crypt.extension "Abracadabra dabracaabrA"
   ```

   where extension is **so** for Solaris, Linux, and HP-UX IA64; and **a** for AIX.
3. Save the file.
4. Restart the Oracle DM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

Creating Network Elements

Important: This is a mandatory configuration task.

Note: If you already created network elements for telephone numbers, you do not need to create them again. The same network elements are used for numbers and SIM cards.

By default, there are only sample network elements. Before creating SIM cards, you need to add the network elements that you will use with your GSM services.

Note: You cannot define network elements by brand; all network elements can be used by any brand.

To customize network elements, you edit the **pin_network_elements** file, then run the "load_pin_network_elements" utility to load the contents of the file into a **/config/network_element** object in the BRM database.

> **Note:** The 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*.

1. Edit the **pin_network_elements** file in *BRM_Home*/**sys/data/config**. The **pin_network_elements** file includes examples and instructions.

   The file includes a list of network elements. For example:

   ```
   sample_network_element_1
   sample_network_element_2
   ```

   > **Caution:** The **load_pin_network_elements** utility overwrites existing network elements. If you are updating network elements, you cannot load new network elements only. You must load complete sets of network elements each time you run the **load_pin_network_elements** utility.

2. Save the **pin_network_elements** file.
3. Use the following command to load the **pin_network_elements** file:

   ```
   load_pin_network_elements pin_network_elements
   ```

To verify that the network elements were loaded, you can display the **/config/network_element** 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*. This example shows a network element in the **/config/network_element** object:

```
PIN_FLD_NETWORK_ELEMENT   STR  [0]  "Sample_Network_Element"
```

Creating SIM Card Formats

> **Important:** This is a mandatory configuration task.

You can customize the list of SIM card formats displayed in SIM Administration Center. The default SIM card formats are **ISO** and **Plug-in**.

> **Note:** You cannot create brand-specific SIM card formats. SIM card formats displayed in SIM Administration Center are available to all brands.

To customize SIM card formats, you edit the **sim_card_types.en_US** sample file in the *BRM_Home*/**sys/msgs/simcardtypes** directory. 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 this command:

```
load_localized_strings sim_card_types.locale
```

Note: 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*.

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

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

Customizing SIM Card Device States

Important: This is a mandatory configuration task.

You can customize SIM card device states to support custom business logic. For example, you can change how SIM cards are pre-provisioned by adding custom device states. See "Customizing How to Pre-Provision SIM Cards".

You can create brand-specific device states. See "Device Management and Brands" in *BRM Developer's Guide*. However, you need to be careful to create device state mappings that work if you change the brand of a SIM card. For example, if a device is in state 2 in one brand, it is still in that state when you change brands, but state 2 might be defined differently.

To customize device states, you edit the **pin_device_state_sim** file in *BRM_Home***/sys/data/config**, and load it by running the **load_pin_device_state** utility. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

If a SIM Card Has No Status Displayed in SIM Administration Center

If a SIM card has no status displayed in SIM Administration Center, the device states for the SIM card's brand have not been loaded. The brand administrator for the brand in question must do the following:

1. Load the **pin_device_state_sim** file for that brand.
2. Load the **sim_device_states.***locale* file if it changed or was not previously loaded.
3. Restart the Connection Manager (CM).
4. Restart the SIM Administration Center.

Adding Device State Names

When you customize device states, you need to add the new device state name to the list of device states that is displayed in SIM Administration Center. To do so, you edit the **sim_device_states.en_US** sample file in the *BRM_Home***/sys/msgs/simdevicestates** directory. 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 this command:

```
load_localized_strings sim_device_states.locale
```

Note: 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*.

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

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

Customizing Service Types

Important: This is a mandatory configuration task.

You can change the services that are allowed to be associated with SIM card devices. For example, you can associate customized services or restrict the list of services.

To change service types, edit the **pin_device_permit_map_sim** file and load it by running the **load_pin_device_permit_map** utility. See "Defining Device-to-Service Associations" in *BRM Developer's Guide*.

Customizing How to Pre-Provision SIM Cards

Important: This is a mandatory configuration task.

You can customize pre-provisioning as follows:

- You can bypass pre-provisioning. To do so, change the device state transition by editing the **pin_device_state_sim** file and loading it with the **load_pin_device_state** utility.

  This example shows how to configure SIM card device states without provisioning:

```
# The storable class to be loaded
/config/device_state/sim
# The type of device for this config file
/device/sim
# Raw -> Released
0: 0: 0: 1: 0: 0
         2: 0: 0
# Release -> Assigned
2: 2: 2: 1: 0: 0
         5: 0: 0
# Assigned -> Unassigned
5: 2: 5: 1: 0: 0
         6: 0: 0
# Unassigned -> Unassigned
6: 3: 6: 1: 0: 0
```

  See "Customizing SIM Card Device States".

- You can change the default pre-provisioning service to use your custom pre-provisioning service. See "Specifying the Pre-Provisioning Service". You can also change the dummy MSISDN used for pre-provisioning. See "Changing the Pre-Provisioning MSISDN".

Specifying the Pre-Provisioning Service

By default, there is no pre-provisioning service. If you pre-provision SIM cards, you must create a pre-provisioning service object so you can associate pre-provisioned SIM cards with a service.

When you pre-provision SIM cards, all SIM cards (or all SIM cards in a brand) are associated with a service by using a pre-provisioning service object. When you assign a SIM card to an account, the SIM card is associated with the customer's service object. (See "About Associating a SIM Card with a Customer's Service".)

To change the pre-provisioning service, you edit the **pin_sim_config** file and run the **load_pin_sim_config** utility to load the contents of the file into the **/config/pre_provisioning_sim** object in the BRM database.

> **Important:** The **load_pin_sim_config** utility needs a configuration file in the directory from which you run the utility. The login that you enter in the configuration file specifies the brand that the SIM card pre-provisioning service applies to. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide* and "Running Utilities with a Branded Database" in *BRM Managing Customers*.

1. To use a different service, you need the POID of a service object created by using your custom service subclass:
   a. (Optional) If you use a custom service, create the service. See "Adding Support for a New Service" in *BRM Developer's Guide*.
   b. Create a plan that uses the service: for example, **/service/telco/gsm/telephony**: and create an account using that service. You can use the **root** account.
   c. Use the **testnap** application or the Object Browser to find the POID of the service object.

      If you use the **testnap** application, use the **robj** command. See "Reading an Object and Writing Its Contents to a File" in *BRM Developer's Guide*.
2. Edit the **pin_sim_config** file in *BRM_Home***/sys/data/config**:
   - Add the POID of the default pre-provisioning service.
   - Remove the comment character (#) from the beginning of the line.

   The default entry is:

   ```
   # Service_Poid:0.0.0.1 /service/telco/gsm/telephony 0 0
   ```

   Example of a customized entry:

   ```
   Service_Poid:0.0.0.1 /service/telco/gsm/telephony 1 0
   ```
3. Save the **pin_sim_config** file.
4. Use the following command to load the **pin_sim_config** file:

   ```
   load_pin_sim_config pin_sim_config
   ```

For more information, see "load_pin_sim_config".

To verify that the pre-provisioning service POID was loaded, you can display the **/config/pre_provisioning_sim** 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*.) The **/config/pre_provisioning_sim** object shows the pre-provisioning service in the PIN_FLD_SERVICE_OBJ field:

```
1  PIN_FLD_SERVICE_OBJ  POID [0] 0.0.0.1 /service/telco/gsm/telephony 1 0
```

Changing the Pre-Provisioning MSISDN

To change the MSISDN used for pre-provisioning, change the following value in the **fm_prov_wireless_pol_create_svc_order.c** policy source code file and recompile it:

```
#define msisdn_val "4085551212"
```

For more information, see "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.

Loading Order Status Definitions

Important: This is a mandatory configuration task.

The status of SIM card orders is displayed in SIM Administration Center. You cannot change the functionality of the order status: for example, you cannot add a custom status: however, you can localize the status definitions that are displayed.

Important: Even if you do not use localized order status definitions, you need to load the **order_sim_status.***locale* file.

To localize the file, you edit a copy of the **order_sim_status.en_US** sample file in the *BRM_Home***/sys/msgs/ordersimstatus** directory and save the edited version with the correct locale file extension. To load the file, you 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 this command:

```
load_localized_strings order_sim_status.locale
```

Note: 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*.

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

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

Customizing SIM Administration Center

You can edit the **SIMAdmin.properties** file to customize how SIM Administration Center works. See the following topics:

- Specifying the Maximum Quantity of SIM Cards in a Request File
- Specifying the Maximum Quantity of SIM Cards in an Order
- Specifying the Information You Want to Receive about SIM Cards
- Specifying whether the Response File Includes a Check Digit

Specifying the Maximum Quantity of SIM Cards in a Request File

You can customize the quantity of SIM cards per vendor request file. By default, the quantity is 5,000. Contact your SIM card vendor to find out if they have a maximum quantity of SIM cards per file.

1. Open the **SIMAdmin.properties** file in the following folder:

   C:\Program Files\Common Files\Portal Software

2. Edit the **RequestFileCardsPerBatch** entry:

   Important: The value must be a multiple of 50.

   ```
   RequestFileCardsPerBatch=5000
   ```

3. Save and close the file.
4. Restart SIM Administration Center.

Specifying the Maximum Quantity of SIM Cards in an Order

By default, the maximum quantity of SIM cards in a single order is 499,995,000.

1. Open the **SIMAdmin.properties** file in the following folder:

   C:\Program Files\Common Files\Portal Software

2. Edit the **MaxOrderQuantity** entry:

   ```
   MaxOrderQuantity=499995000
   ```

3. Save and close the file.
4. Restart SIM Administration Center.

Specifying the Information You Want to Receive about SIM Cards

The information returned about SIM cards from the vendor is specified in the **Var_Out** field in the vendor request file. By default, this field is formatted in the request files as follows:

```
Var_Out:PIN1/PIN2/PUK1/PUK2/ADM1/Ki
```

To change how the entry is formatted in the request files:

1. Open the **SIMAdmin.properties** file in the following folder:

   C:\Program Files\Common Files\Portal Software

2. Edit the **Var_Out** entry:

   ```
   Var_Out=PIN1/PIN2/PUK1/PUK2/ADM1/Ki
   ```

3. Save and close the file.
4. Restart SIM Administration Center.

Specifying whether the Response File Includes a Check Digit

In some cases, the SIM card vendor requires a check digit in the response file. By default, SIM Administration Center assumes that the vendor response file does not include a check digit.

To include a check digit in the response file:

1. Open the **SIMAdmin.properties** file in the following folder:

   C:\Program Files\Common Files\Portal Software

2. Edit the **ResponseContainsCheckDigit** entry:

   ```
   ResponseContainsCheckDigit=false
   ```

3. Save and close the file.
4. Restart SIM Administration Center.

Specifying the Size of Searches

You can change the value of the step search that SIM Administration Center uses. For slower systems, a smaller step search size can return results faster.

1. Open the **SIMAdmin.properties** file in the following folder:

   C:\Program Files\Common Files\Portal Software

2. Edit the **searchStepSize** entry:

   ```
   searchStepSize=100
   ```

3. Save and close the file.
4. Restart SIM Administration Center.

Specifying SIM and IMSI Number Maximum Length

You can determine whether SIM Administration Center prepends zeroes to SIM and IMSI numbers to make them the maximum standard length (19 characters for SIM and 15 for IMSI). This property applies to numbers created in request files and to numbers specified in searches in SIM Administration Center.

By default, zeroes are not prepended to the numbers. Setting the **maximumStandardLength** property to **true** causes SIM Administration Center to prepend zeroes to SIM and IMSI numbers.

> **Note:** In previous versions, the default behavior was to prepend zeroes to the numbers. Therefore, if you need to search on SIM card inventory created in earlier versions, you should set the **maximumStandardLength** property to **true**.

1. Open the **SIMAdmin.properties** file in the following folder:

C:\Program Files\Common Files\Portal Software

2. Edit the **maximumStandardLength** entry:

```
maximumStandardLength=true
```

3. Save and close the file.
4. Restart SIM Administration Center.

Creating SIM Cards for Testing

To test your wireless integration, you need SIM card devices in your database so you can create accounts. You can create a set of SIM cards to test with.

> **Note:** This procedure creates an order with five SIM cards. The standard file format uses multiples of 50 SIM cards.

To create a set of SIM cards to test with:

1. In SIM Administration Center, create an order using the following entries:
   - Customer Info tab:

     Customer name: Sample

     Address: 100 Main St

     City: Anytown

     State/Province: CA

     Zip/Postal: 99999

     Country: USA
   - Order Details tab:

     Quantity: 5

     SIM prefix: 1000

     Start SIM: 3000

     IMSI prefix: 1000

     Start IMSI: 4000

     SIM card format: Plug-in

     Transport key: 255
2. Generate the request file.
3. Open the request file in a text editor. The file name is **Sam00001.inp**.
4. Save the file as **Sam00001.out**. This file will be the vendor response file.
5. Add the following lines after the last line in the file:

```
10004000 10003000 7349 6215 37177972 11267846 83307381
1A73DFCD69F6E11DF3678DC81C9E8057
10004001 10003001 6604 3374 84688421 63029874 91983041
B6FB4A9029FFB2139A2828CF541A566B
10004002 10003002 3708 9818 35042443 24656819 46904714
BB5EA7413C9C237DE94ECC6A55ECA7A6
```

```
10004003 10003003 4856 6049 92238513 12035890 92454872
B2888B7F6471501D4AFA1E6D67853581
10004004 10003004 3315 5878 94466890 55544514 19668055
7EC49FD7144221295F2E1090AD92D623
```

6. Save the file.
7. Load the response file.

Uninstalling SIM Manager

To uninstall SIM Manager on UNIX, run the *BRM_Home*/**uninstaller/SIMMgr/uninstaller.bin**.

38

load_pin_sim_config

Use this Oracle Communications Billing and Revenue Management (BRM) utility to specify the SIM card pre-provisioning service. You can use different pre-provisioning services for each brand.

For more information, see:

- Specifying the Pre-Provisioning Service
- About SIM Card Pre-Provisioning

> **Note:** You cannot load separate **/config/pre_provisioning_sim** objects for each brand. All brands use the same object.

> **Important:** To connect to the BRM database, the **load_pin_sim_config** 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*. The login that you enter in the configuration file specifies the brand that the pre-provisioning service applies to. See "Running Utilities with a Branded Database" in *BRM Managing Customers*.

- About Managing Telephone Numbers

Location

*BRM_Home***/bin**

Syntax

```
load_pin_sim_config [-d] [-v] file_name
```

Parameters

-d

Enables debugging mode.

-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_sim_config *any_other_parameter* **–v >** *filename***.log**

file_name

The file, for example, **pin_sim_config** that includes the value to use for the pre-provisioning service.

Results

Reports success or displays an error.

Part IX

Managing Voucher Inventory

Part IX describes how to manage voucher inventory in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

39

About Managing Voucher Inventory

This chapter describes how to manage voucher inventory in your Oracle Communications Billing and Revenue Management (BRM) system by using Voucher Manager and Voucher Administration Center.

Important: Voucher Manager and Voucher Administration Center are optional components, not part of base BRM.

Before using Voucher Manager and Voucher Administration Center, you need to know the following information:

- Basic BRM concepts. See *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- How to create and edit BRM configuration files. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

About Managing Voucher Cards

You use Voucher Manager and Voucher Administration Center to manage the life cycle of voucher cards.

- Voucher Manager includes the opcodes and storable classes required for creating and managing voucher card devices in the BRM database.
- Voucher Administration Center is the client application that you use to create and process voucher orders and voucher brands.

You use Customer Center to assign voucher cards to customers and manage voucher cards in customer accounts. See "Getting Information about Voucher Usage".

About Creating Vouchers

Use Voucher Administration Center to order voucher cards from a vendor. For complete instructions on using Voucher Administration Center, see the Voucher Administration Center Help.

Important:

- Before you create an order, you must configure voucher data. See "Mandatory Configuration Tasks". You must also create voucher deals in Pricing Center. See "About Voucher Deals".
- When creating orders, do not log on to Voucher Administration Center as **root**. If you do, Voucher Administration Center will stop responding when you try to select any deal, other than Product Purchase Fee Event, to retrieve deal information for the order. Instead, log on as a CSR user.

To create vouchers:

1. Create an order in Voucher Administration Center.

   The order is stored in the BRM database and can be updated or canceled. See "About Voucher Orders".

2. Create one or more *vendor request files*. See "Vendor Request Files and Voucher Orders".

3. Send the vendor request files to the vendor.

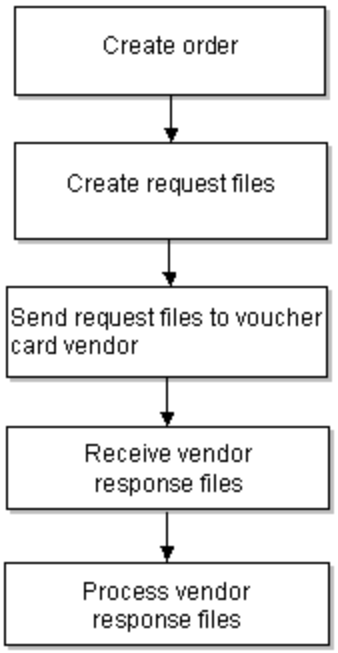
   You can use any file transfer method, such as FTP or email.

4. Receive the *vendor response files* and process them in Voucher Administration Center. See "Vendor Response Files and Voucher Devices".

   This creates an **/order/voucher** object for each voucher card.

Figure 39–1 shows the process for creating voucher cards:

Figure 39–1 Voucher Card Creation Process



About Voucher Orders

When you create an order, you can specify the following order attributes:

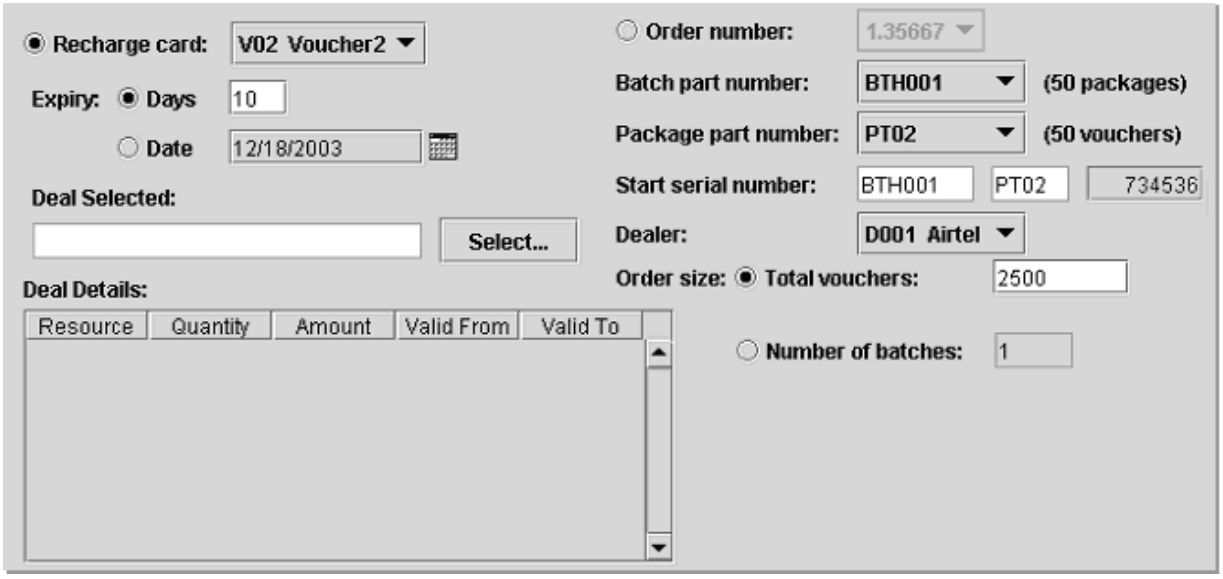
- Customer name and address information. This is usually your company.
- Vendor name and address of the vendor who creates the vouchers.

> **Note:** There are limitations on how many characters you can use for each piece of customer or vendor information. These limitations allow more characters than you will probably use. For more information, see the **/order** object definition.

- Voucher card attributes:
  - Quantity of voucher cards
  - Package and batch part numbers and quantity
  - Voucher card number
  - Brand
  - Deal associated with the voucher
  - Voucher card expiration date
  - Dealer name

Figure 39–2 shows the **Order Details** tab you use to specify voucher order attributes in Voucher Administration Center:

Figure 39–2 Order Details Tab



About Voucher Deals

Customers purchase vouchers by purchasing a voucher deal. You create a voucher deal in Pricing Center, following the normal procedure. The deal must have the following attributes:

- There must be only one product in the deal.

- There must be only one balance impact in the product.
- The product must rate only a product purchase fee event.
- The product must be an item product.

For information about creating deals, see "About Deals" in *BRM Setting Up Pricing and Rating*.

About Dealers

Dealers are the voucher card distributors from whom you buy the voucher cards. A dealer can also be the service provider. You must load the dealers details (dealer name and dealer code) before you start using Voucher Manager and Voucher Administration Center.

The dealer codes are used by BRM system and dealer names are displayed in Voucher Administration Center. You can also search for dealer names, and display them in reports.

Use the **load_pin_dealers** utility to load the dealer information into the BRM database. See "Loading Dealer Details".

About Recharge Card Types

You must create recharge card types and load them into the database. For example, a card type might be *$20 Promotional Card*. To create a card type, you must enter the recharge card details:

- Dealer name.
- Dealer code.
- Recharge card type (a description).
- Recharge card code.

The recharge codes are used by BRM system and recharge card types are displayed in Voucher Administration Center.

You must load the recharge card details before you start using Voucher Manager and Voucher Administration Center. Use the **pin_recharge_card_type** utility to load the recharge card information into the BRM database. See "Loading Recharge Card Details".

About Voucher Details

You must define voucher details such as batch part number, package part number, package quantity, and batch quantity before using Voucher Manager and Voucher Administration Center:

- A *package* is a set of vouchers with the same recharge amount.
- A *batch* is a carton containing several packages of vouchers.

The batch part number and package part number are used to create the voucher card number. The unique serial number for each voucher card is the concatenation of the batch number, the package part number, and the voucher card number. For example, if you enter 0002 for the batch number, 0003 for the package part number, and 1000 for the starting number, the unique serial number would be 000200031000.

Use the **load_pin_voucher_config** utility to load the voucher information into the BRM database. See "Loading Voucher Details".

About Managing Orders

When you create and manage voucher orders, BRM assigns a state to the order. You use Voucher Administration Center to display the status of an order.

> **Note:** Voucher orders are stored in the database as **/order/voucher** objects.

Voucher Manager sets an order's status as follows:

1. When you create an order, the status is *New*. You can save a new order or cancel it. When you cancel an order, the status is set to *Cancelled*. You can also update an order if it has a status of *New*.
2. After you create the vendor request file, the status is changed to *Request*. When an order has the *Request* status, you cannot cancel or modify it.
3. When you begin processing vendor response files, the status is changed to *Partial Receive*.
4. After you process all the vendor response files in an order, the status is changed to *Received*.

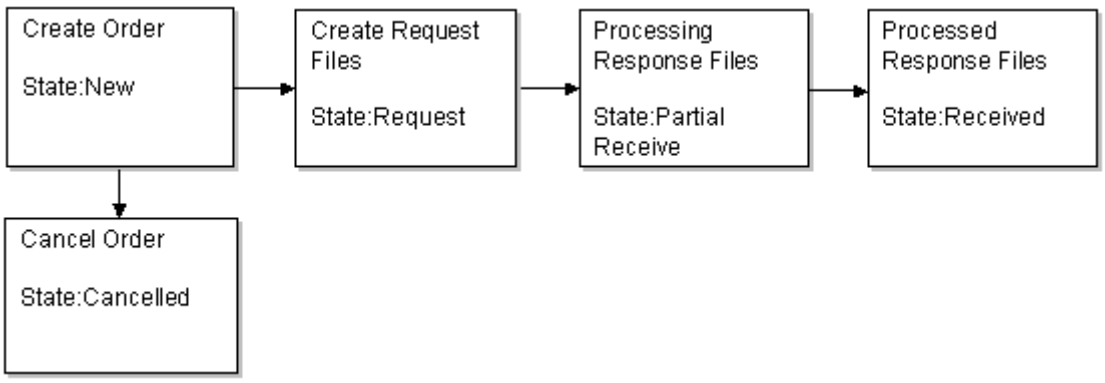
   When the order has a Received or Partial Receive status, you cannot change the order attributes. You can search for orders of any status.

The voucher order states are represented as numbers in Voucher Administration Center.

- 1 = New
- 2 = Request
- 3 = Received
- 4 = Partial Receive
- 5 = Cancelled

Figure 39–3 shows the various order states:

Figure 39–3 Order States



About Modifying Orders

You use Voucher Administration Center to modify an order. You can search for orders by entering a date range, order ID, recharge card type, dealer name, or order state

value. For instructions on modifying vouchers, see the Voucher Administration Center Help.

About Voucher Request and Response Files

You manage orders with your voucher vendor by exchanging *request* and *response* files.

- The request files include information about the order, and are used by the vendor to create the vouchers. *Request* files specify the quantity of vouchers and the voucher attributes, such as the customer name and voucher format.
- The response files contain information from the vendor about the vouchers. You load this data into the BRM database.

> **Note:** Before you create request files or upload response files, you must create vendor request and response file templates. See "About Request and Response File Templates".

Before creating the request file, you must set the configuration values. The configuration information includes directory path name, the standard naming for the request file, as well as the encryption algorithm.

By default the voucher PIN is encrypted when it is stored in the database. An encrypted request file consists of the serial number, PIN, recharge type, dealer name, quantity, order creation date, order reference number as well as sender and receiver contact information.

> **Note:** Multiple vendor request files are created automatically, based on the size of the order. However, you need to process each vendor *response* file individually.

Vendor Request Files and Voucher Orders

Voucher Administration Center generates vendor request files using the following naming convention:

*CustomerNamennnnn***.inp**

where:

- *CustomerName* is the name in the **Customer name** field when creating the order.
- *nnnnn* specifies the order in which the vendor request files are generated. The number is incremented for each vendor request file generated by an order. For example, the first vendor request file in an order is 00001.

For example, if the customer name is Oracle, the first vendor request file in the order is named **Oracle00001.inp**.

> **Important:** All orders that you create for the same customer use the same file names. Therefore, you should always put request files in a new or empty folder for each order, so files from previous orders are not overwritten.

Vendor request files cannot specify more than 5,000 vouchers. If you create an order for more than 5,000 vouchers, the order is automatically divided into multiple files.

For example, if you create an order for 50,000 vouchers, Voucher Administration Center creates 10 vendor request files of 5,000 vouchers each.

You can create a maximum order of 499,995,000 vouchers, which would create 99,999 vendor request files.

You use Voucher Manager to:

- Update the status of the order from *New* to *Request*.
- Set the voucher expiration date.
- Set the request file name and creation date.

You can change the default values for the maximum quantity of vouchers in an order and the maximum number of vouchers in a request file.

Vendor Response Files and Voucher Devices

Vendor response files use the following naming convention:

*CustomerName_nnnnn***.out**

where:

- *CustomerName* is the name in the **Customer name** field when creating the order.
- *nnnnn* is the file order number, which is incremented for each order. The first file in an order uses 00001.

For example, if the customer name is "Oracle," the first file in an order is named **Oracle00001.out**.

You use Voucher Administration Center to process the vendor response files and create devices.

After the vendor response file is processed, the state of the order is changed to Received. In case a partial response file is received, the order is state is changed to Partial Receive.

The voucher device created by this process consists of the voucher pin, voucher serial number, dealer name, expiration date, and status. The various states of voucher devices are new, used, and expired. See "About Voucher Device States".

You can modify voucher devices. See "About Managing Vouchers".

About Request and Response File Templates

To process vendor request and response files from various vendors, you create vendor file templates. These templates define the request and response file formats for each vendor. You typically use one template for each vendor. A basic format is available in the BRM database, and you can create a new template or open an existing template and modify it.

You create templates by using Voucher Administration Center, and you specify a template when generating the request and response files. The templates are stored in XML format in the BRM database as **/config/inventory_mgmt_template** objects.

Figure 39–4 shows a sample request template:

Figure 39–4 Sample Request Template

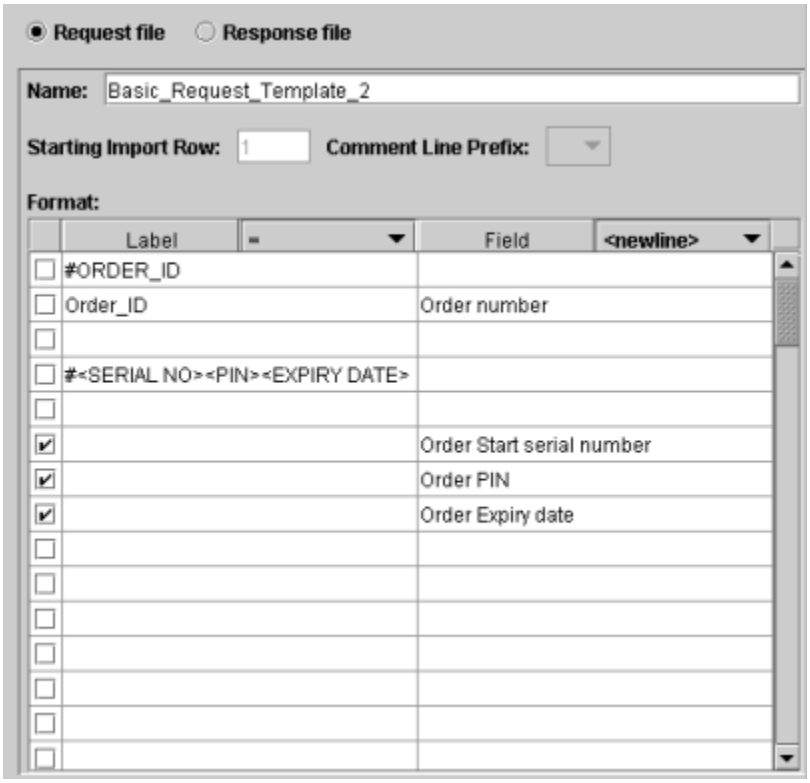


Figure 39–5 shows a sample response template:

Figure 39–5 Sample Response Template

About Request and Response File Encryption

You can encrypt the vendor request file, and you can decrypt the vendor response file if your vendor returns response files that are encrypted. Before encrypting files, you need to specify the request and response encryption type in the **EncryptionResources.properties** file.

This file is located in the path: **C:\Program Files\Portal Software\VoucherAdministrationCenter\encrypt.jar**.

About Managing Vouchers

After vouchers are created, you assign them to customer accounts by using Customer Center. See information about topping up accounts in the Customer Center Help.

You can search for a single voucher, multiple vouchers or brand accounts and then modify the voucher's state, expiration date, or brand attributes. You can also *t*ransfer the amount from the voucher to the customer's account.

About Voucher Device States

Voucher devices have the following device states:

- When you process a vendor response file, you create voucher devices in the database. The device state of the vouchers is set to New.

- In Customer Center, when an account is created and assigned a voucher, the device state is set to *Used*.
- If an account is inactive or if an amount is transferred from a voucher device to customer's account, the device state is *Used*.
- When the voucher is used up or expires, the device state is set to *Expired*.

Voucher devices have the following states:

- 1 = New
- 2 = Used
- 3 = Expired

The voucher states are represented with its related number in Voucher Administration Center.

About Managing Vouchers in a Branded System

When you log in to Voucher Administration Center, your login associates you with a brand. When you create and process an order, you can assign the order to the login brand, or any sub-brand. All vouchers in the order are also associated with the same brand.

Important: Use Voucher Administration Center to update the brand of a device or an order when their state is *New*.

For information about branding, see "About Branding" in *BRM Managing Customers*.

Managing Vouchers in a Multischema System

If you manage vouchers in a multischema system:

- In a multischema environment that is not brand enabled, you can create vouchers only in the database schema that you are logged in to.
- If you log in as a brand host administrator and select Brand Host as the brand to create an order, you can create orders only in the schema that you logged in to. If you select any other brand, the order is created in the schema that hosts the selected brand.
- You cannot change the brand of a voucher to a brand that is hosted in a different schema from the schema that the current brand is hosted in. For example, if a voucher belonging to Brand A is in schema 0.0.0.1, you cannot change the voucher to Brand B if Brand B is in schema 0.0.0.2.
- When searching for vouchers or orders, if you log in as a brand host administrator and you select the top-level brand without including sub-brands in the search criteria, the search results include only vouchers or orders from the schema you logged in to.

Managing Expired Vouchers

If a voucher is not used, it expires. You must use the **pin_voucher_expiration** utility to change the state of the vouchers to Expired.

Note: The **pin_voucher_expiration** 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*.

Use the following command to run the **pin_voucher_expiration** utility:

```
pin_voucher_expiration [-v] [-d]
```

For more information, see "pin_voucher_expiration".

About Voucher Customization Options

You can use policy opcodes to customize how vouchers are created and managed. See "Voucher Manager FM Standard Opcodes" in *BRM Developer's Reference*.

You can choose to encrypt the vendor request file and decrypt the vendor response file if your vendor returns response files that are encrypted. See "About Request and Response File Encryption".

Getting Information about Voucher Usage

You can get information about voucher inventory and status by running the Voucher Management reports. See "Voucher Manager Reports" in *BRM Reports*.

How Voucher Association Works

PCM_OP_VOUCHER_ASSOCIATE_VOUCHER calls PCM_OP_DEVICE_ASSOCIATE to perform these operations:

- Calculate the balance impacts of purchasing the deal linked to the voucher device (**/device/voucher** object).
- Associate a voucher device with an account or a service.

Before calling the device association opcode, this opcode uses the voucher device ID and PIN specified in the input flist to find the voucher device POID. It passes this POID to the device association opcode.

During voucher top-up operations, the PCM_OP_PYMT_TOPUP opcode calls the PCM_OP_PYMT_POL_VALID_VOUCHER policy opcode, which in turn calls PCM_OP_VOUCHER_ASSOCIATE_VOUCHER. PCM_OP_VOUCHER_ASSOCIATE_VOUCHER reformats the information it receives from the device association opcode so that it can be used by the top-up opcode. See "Performing Top-Ups with PCM_OP_PYMT_TOPUP" in *BRM Configuring and Collecting Payments*.

If the device association succeeds, PCM_OP_VOUCHER_ASSOCIATE_VOUCHER returns all currency and non-currency balance impacts of the voucher.

If the device association fails, the opcode returns an error in the error buffer.

PCM_OP_VOUCHER_ASSOCIATE_VOUCHER returns an error under these circumstances:

- It cannot find a voucher device associated with the input ID and PIN in the BRM database.
- The device cannot be associated with the account or service.

Customizing How Voucher Manager Manages Devices

Use the following opcodes to customize Voucher Manager devices:

- PCM_OP_VOUCHER_POL_DEVICE_CREATE. See "Customizing Voucher Creation".
- PCM_OP_PYMT_POL_VALID_VOUCHER. See "Customizing Voucher Validation".
- PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE. See "Customizing Voucher Association".
- PCM_OP_VOUCHER_POL_DEVICE_SET_ATTR. See "Customizing Voucher/Service Association".
- PCM_OP_VOUCHER_POL_DEVICE_SET_BRAND. See "Setting the Brand for a Voucher".

Customizing Voucher Creation

PCM_OP_VOUCHER_POL_DEVICE_CREATE validates a device during device creation. For example, this policy opcode verifies that the numbers have the correct number of digits and use the proper syntax. It also verifies that the voucher does not already exist in the database.

You can customize this opcode to change the validation rules for creating voucher devices.

This opcode is called by PCM_OP_DEVICE_CREATE when creating a voucher device.

PCM_OP_VOUCHER_POL_DEVICE_CREATE returns the same values that are in the input flist. Some values might be reformatted, for example, extra spaces might be deleted.

This opcode returns an error when any of the values are not valid or when the voucher already exists in the database.

Customizing Voucher Validation

BRM uses the PCM_OP_PYMT_POL_VALID_VOUCHER policy opcode to validate vouchers. By default, this policy opcode interacts with Voucher Manager. If you use a third-party voucher management system, you must customize this policy opcode to work with that system. Vouchers are generally used for top-ups and prepayments only.

PCM_OP_PYMT_POL_VALID_VOUCHER is called by PCM_OP_PYMT_TOPUP during voucher top-up operations. To interact with a voucher management system, this policy opcode calls PCM_OP_VOUCHER_ASSOCIATE_VOUCHER. See "Performing Top-Ups with PCM_OP_PYMT_TOPUP" in *BRM Configuring and Collecting Payments*.

Customizing Voucher Association

PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE calculates the balance impacts of associating a voucher device (**/device/voucher** object) with an account or a service.

This policy opcode is called by the PCM_OP_DEVICE_ASSOCIATE opcode as follows:

- To retrieve the balance impacts but *not* the information that enables PCM_OP_DEVICE_ASSOCIATE to associate the voucher with an account or a service, it is called with the PCM_OPFLG_CALC_ONLY flag *on* (calculation-only mode).

- To retrieve the balance impacts *and* the information that enables PCM_OP_DEVICE_ASSOCIATE to associate the voucher with an account or a service, it is called with the PCM_OPFLG_CALC_ONLY flag *off*.

If the voucher is not expired, PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE calls PCM_OP_SUBSCRIPTION_PURCHASE_DEAL to calculate the currency and non-currency balance impacts of the deal linked to the voucher.

> **Note:** Whether the PCM_OPFLG_CALC_ONLY flag is on or off, this policy opcode always calls PCM_OP_SUBSCRIPTION_PURCHASE in calculation-only mode. During top-up operations, this enables PCM_OP_PYMT_TOPUP to perform the actual balance impacts.

In addition, when called with the PCM_OPFLG_CALC_ONLY flag *off*, PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE calls PCM_OP_DEVICE_SET_STATE to update the voucher device status as follows:

- If the voucher expiration date is passed, updates it from **new** to **expired**.
- If the voucher is not expired, updates it from **new** to **used**.

If successful, PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE returns the following information:

- Balance impacts of purchasing the deal linked to the voucher device
- Validity dates of the resources associated with the affected balances

PCM_OP_VOUCHER_POL_DEVICE_ASSOCIATE returns an error in the following cases:

- The device and the account or service are not in the same brand.
- The voucher PIN specified in the input flist is not in the BRM database.
- The specified voucher is expired.
- The specified voucher is used.

Customizing Voucher/Service Association

PCM_OP_VOUCHER_POL_DEVICE_SET_ATTR ensures that the voucher card number (PIN_FLD_DEVICE_ID) cannot be changed during a device update. It validates a deal object available in the database if the deal object is changed.

You can customize this opcode to change how vouchers are associated with services.

This opcode is called by PCM_OP_DEVICE_SET_ATTR when updating a voucher card device.

PCM_OP_VOUCHER_POL_DEVICE_SET_ATTR returns an error if an attempt is made to change voucher pin or device ID, and when a deal object is changed.

Setting the Brand for a Voucher

PCM_OP_VOUCHER_POL_DEVICE_SET_BRAND validates that the voucher device state is New, when changing the voucher brand.

You can customize this opcode to change how vouchers can be associated with brands.

This opcode returns an error if the device state is not New.

Customizing How Voucher Manager Manages Orders

Use the following opcodes to customize Voucher Manager orders:

- PCM_OP_VOUCHER_POL_ORDER_CREATE. See "Customizing Order Creation".
- PCM_OP_VOUCHER_POL_ORDER_ASSOCIATE. See "Customizing Order Association".
- PCM_OP_VOUCHER_POL_ORDER_SET_ATTR. See "Customizing Order Attributes".
- PCM_OP_VOUCHER_POL_ORDER_SET_BRAND. See "Setting the Brand for an Order".
- PCM_OP_VOUCHER_POL_ORDER_PROCESS. See "Canceling Orders".
- PCM_OP_VOUCHER_POL_ORDER_DELETE. See "Deleting Orders".

Customizing Order Creation

PCM_OP_VOUCHER_POL_ORDER_CREATE validates the information in the input flist before an order object is created.

Validation consists of:

- Checking for duplicate serial numbers.
- Verifying the quantity ordered.

  When calculating the quantity of vouchers within each order the following formula is applied:

  Quantity = batch quantity * package quantity * total batches

  Note: * indicates multiplication.

- Checking that the quantity is same as the quantity specified in the order.
- Checking for the deal object existence.

You can customize this opcode to change the validation rules for creating **/order/voucher** objects.

This policy opcode is called by PCM_OP_ORDER_POL_CREATE when the **/order/voucher** object is created.

If any of the checks fail, this opcode returns an error message and logs an error in the CM **pinlog** file, indicating the reason for the failure. The **/order/voucher** object creation is terminated.

Customizing Order Association

PCM_OP_VOUCHER_POL_ORDER_ASSOCIATE ensures that a sub-order cannot be associated or disassociated with the master order when the order state is not New.

You can customize this opcode to change any validation for voucher order association.

This policy opcode is called by PCM_OP_ORDER_POL_ASSOCIATE when associating or disassociating a sub-order with the master order at account creation.

If the opcode is successful:

- In case of association, the device is associated with the account or service and the device state is set to **Assigned**.
- In case of disassociation, the device is disassociated with the account or service and the device state is set to **Unchanged**.

This opcode returns an error when you associate or disassociate a sub-order with the master order when the order state is not **New**.

Customizing Order Attributes

PCM_OP_VOUCHER_POL_ORDER_SET_ATTR validates the new values passed into the input flist before an order is modified.

This policy opcode is called by PCM_OP_ORDER_POL_SET_ATTR when updating the attributes and description of the **/order/voucher** object.

This policy opcode verifies that:

- The order has not yet been sent to the vendor.
- The modified serial number has no duplicate number in the database.
- The order quantity is correct.

  When calculating the quantity of vouchers within each order the following formula is applied:

  Quantity = batch quantity * package quantity * total batches

  Note: * indicates multiplication.

- The order is not modified when it is in the **Received** or **Canceled** state.

During validation, if it is found that the order was not yet sent to the vendor, the values are validated and updated with the new values passed in.

During validation, if it is found that the order has been sent to the vendor, updating or modifying the order is not allowed.

Setting the Brand for an Order

PCM_OP_VOUCHER_POL_ORDER_SET_BRAND ensures that the brand of an order cannot be changed when the order state is in **Request** or **Partial Receive**.

This policy opcode is called by PCM_OP_ORDER_POL_SET_BRAND when changing the brand association of an **/order/voucher** object.

This opcode returns an error message if a change is attempted on the order in **Received** or **Partial Receive** state.

Canceling Orders

PCM_OP_VOUCHER_POL_ORDER_PROCESS reads the status of an order using the order POID and terminates the processing of the order if the order state is **Cancel**.

A request file is sent to the relevant vendor and the vendor returns a vendor response file. The PCM_OP_ORDER_POL_PROCESS calls this opcode while processing the vendor response files.

Deleting Orders

PCM_OP_VOUCHER_POL_ORDER_DELETE ensures that an order cannot be deleted when the order is in the **Received** or **Partial Receive** state.

This policy opcode is called by PCM_OP_ORDER_POL_DELETE when deleting an order.

If deletion is attempted on the order object in **Request** or **Partial Receive** state, this opcode returns an error.

40

Installing and Configuring Voucher Manager and Voucher Administration Center

This chapter describes how to install and customize Oracle Communications Billing and Revenue Management (BRM) Voucher Manager and Voucher Administration Center.

For information about Voucher Manager and Voucher Administration Center, see "About Managing Voucher Inventory".

Hardware and Software Requirements

Before installing Voucher Manager, you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in *BRM Installation Guide*.
- BRM.
- Oracle 10g or Oracle 11g.

Before installing Voucher Administration Center, you must install:

- BRM.

Voucher Manager is supported on the HP-UX IA64, Linux, Solaris, and AIX operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

Note: Voucher Administration Center requires the Java Runtime Environment (JRE). It is included in the Voucher Administration Center package and is approximately 50 MB. If the JRE was already installed with another BRM client application, it will not be reinstalled.

Mandatory Configuration Tasks

In addition to installing the Voucher Manager software, you must configure and load several text files before you can use Voucher Manager and Voucher Administration Center.

Before performing these configuration tasks, you should gather all the information about your voucher dealers, your recharge card types, and the batch and package numbers you will use.

Note: You can perform these tasks in any order, but you must install Voucher Manager first.

- Define voucher device states. This involves two tasks:
  - Define and load the state transition model.
  - Load localized state descriptions.

  See "Customizing Voucher Device States" and "Loading Voucher Device States".
- Define order device states. See "Loading Voucher Order States".
- Specify the services with which vouchers can be associated and load this information into the BRM database. See "Customizing and Loading Voucher Service Associations".
- Associate the voucher devices with batch and part numbers and with batch and package quantities. See "Loading Voucher Details".
- Define your voucher dealers and load this information into the database. You can also associate dealers with a specific voucher recharge card type. See "Loading Dealer Details".
- Define your recharge card types. Each recharge card type is associated with an order configuration, which includes voucher dealer information, batch number, and order size. See "Loading Recharge Card Details".
- Load the following locale files to localize the description fields in Voucher Administration Center:
  - To localize voucher device state descriptions, load the **device_state_voucher.***locale* file located in *BRM_Home*/**sys/msgs/voucher_devicestates**. Even if you do not run a localized version of Voucher Administration Center, you need to load this file.
  - To localize voucher order state descriptions, load the **order_state_voucher.***locale* file located in *BRM_Home*/**sys/msgs/voucher_orderstates**. Even if you do not run a localized version of Voucher Administration Center, you need to load this file.

Installing Voucher 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 Voucher 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 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.

Caution: 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 the temp_dir directory and enter this command:

```
7.5.0_VoucherMgr_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. (Optional) To install Voucher Manager server components separately on this computer or on another computer, select custom install when asked to specify the setup type. Select the components you are installing by typing their respective numbers and click **Next**. The components are:
   - **Voucher Manager**
   - **Order Manager**

   Follow the instructions displayed during installation. The default installation directory for Voucher Manager is **opt/portal/7.5**.

Note: The installation program does not prompt you for the installation directory if BRM or Voucher Manager is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the Voucher 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 Voucher Manager installation is now complete.

Installing Voucher Administration Center

To install Voucher Administration Center on Windows:

1. Download the software to a temporary directory *(temp_dir)*.
2. Extract the downloaded **.zip** file to a temporary directory (*temp_dir*).
3. Go to *temp_dir* and run the **setup.exe** program. The installation wizard for Voucher Administration Center starts.
4. Answer the prompts in the installation wizard screens.

Your Voucher Administration Center installation is now complete.

Customizing Voucher Device States

Important: This is a mandatory configuration task.

You can customize voucher device states to support custom business logic.

You can create brand-specific device states. See "Device Management and Brands" in *BRM Developer's Guide*. However, you need to be careful to create device state mappings that work if you change the brand of a voucher. For example, if a device is in state 2 in one brand, it is still in that state when you change brands, but state 2 might be defined differently in a different brand.

To customize voucher device states, you edit the **device_state_voucher.en_US** sample file in the *BRM_Home***/sys/msgs/voucher_devicestates** directory. 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 this command:

```
load_localized_strings device_state_voucher.locale
```

Note: 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*.

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

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

If a Voucher Has No Status Displayed in Voucher Administration Center

If a voucher has no status displayed in Voucher Administration Center, it means that the device states for the voucher's brand have not been loaded. The brand administrator for the brand must do the following:

1. Load the **pin_device_state_voucher** file in *BRM_Home***/sys/data/config** for that brand.
2. Load the **device_state_voucher.***locale* file in *BRM_Home***/sys/msgs/voucher_devicestates** if it changed or was not previously loaded.
3. Restart the Connection Manager (CM).
4. Restart Voucher Administration Center.

Adding Device State Names

When you customize device states, you need to add the new device state name to the list of device states that is displayed in Voucher Administration Center. To do so, you run the **load_localized_strings** utility to load the contents of the **device_state_voucher.***locale* file into **/strings** objects in the BRM database.

> **Note:** *locale* indicates the country. For example, the file used for US English is **device_state_voucher.en_US**.

The **device_state_voucher.***locale* file uses the same format as other files that are loaded by using the **load_localized_strings** utility. See the description of the required format of a string file in "About the String Manipulation Functions" in *BRM Developer's Reference*.

> **Note:** The **load_localized_strings** utility needs a configuration file in the directory from which you run the utility. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

1. Edit the **device_state_voucher.***locale* file in *BRM_Home***/sys/msgs/voucher_devicestates**.

> **Caution:** The **load_localized_strings** utility overwrites all existing notification data in the BRM database. If you are updating device state names, you cannot load new device state names only. You must load all notification data each time you run the **load_localized_strings** utility.

2. Use the following command to run the **load_localized_strings** utility:

```
load_localized_strings device_state_voucher.en_US
```

> **Note:** Enter the appropriate file suffix for other locales. For example, to load device states in the German language, use the file named **device_state_voucher.de**.

Look in the **load_localized_strings.log** file to find any errors. The log file is either in the directory from which the utility was started or in a directory specified in the configuration file.

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

3. If Voucher Administration Center is running, stop and restart it.

To verify that the device state names were loaded, you can display the **/strings** objects 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*.

Loading Order Status Definitions

> **Important:** This is a mandatory configuration task.

The status of voucher orders is displayed in Voucher Administration Center. You cannot change the functionality of the order status; for example, you cannot add a custom status. However, you can localize the status definitions that are displayed.

> **Important:** Even if you do not use localized order status definitions, you need to load the **order_state_voucher.***locale* file.

To localize the file, you edit a copy of the **order_state_voucher.en_US** sample file in the *BRM_Home***/sys/msgs/voucher_orderstates** directory and save the edited version with the correct locale file extension. To load the file, you 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 this command:

```
load_localized_strings order_state_voucher.locale
```

> **Note:** 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*.

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

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

Loading Dealer Details

> **Important:** This is a mandatory configuration task.

To load dealer details into the database, you run the **load_pin_dealers** utility to load the data into the **/config/dealers** object. See "About Dealers".

Important: The 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*.

1. In the **pin_dealers** file in *BRM_Home*/**sys/data/config**, enter the dealer names and code.
2. Save and close the **file.**
3. Use the following command to load the **pin_dealers** file:

   ```
   load_pin_dealers dealer_input_file
   ```

For more information, see "load_pin_dealers".

Loading Recharge Card Details

Important: This is a mandatory configuration task.

To load recharge card details into the database, you run the **load_pin_recharge_card_type** utility to load the data into the **/config/recharge_card_type** object. See "About Recharge Card Types".

Important: The 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*.

1. In the **pin_recharge_card_type file located in** *BRM_Home*/**sys/data/config, enter the recharge card details.**
2. Save and close the **file.**
3. Use the following command to load the **pin_recharge_card_type** file:

   ```
   load_pin_recharge_card_type card_type_input_file
   ```

For more information, see "load_pin_recharge_card_type".

Loading Voucher Device States

Important: This is a mandatory configuration task.

Voucher devices have the following states:

- 1 = New
- 2 = Used
- 3 = Expired

For information on voucher device states, see "About Voucher Device States".

These states are already defined in the **pin_device_state_voucher** file. You run the **load_pin_device_state** utility to load the data into the **/config/device_state/voucher** object.

> **Important:** The **load_pin_device_state** 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*.

1. Edit the **pin_device_state_voucher** file **in** *BRM_Home***/sys/data/config.**
2. Save and close the **file.**
3. Use the following command to load the **pin_device_state** file:

```
load_pin_device_state state_file_device
```

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

Loading Voucher Order States

> **Important:** This is a mandatory configuration task.

Voucher states are defined in the **pin_order_state_voucher** file. You run the **load_pin_order_state** utility to load the data into the **/config/order_state/voucher** object.

For information on voucher order states, see "About Managing Orders".

> **Important:** The **load_pin_order_state** 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*.

1. Edit the **pin_order_state_voucher** file **in** *BRM_Home***/sys/data/config.**
2. Save and close the **file.**
3. Use the following command to load the **pin_order_state** file:

```
load_pin_order_state state_file_order
```

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

Customizing and Loading Voucher Service Associations

> **Important:** This is a mandatory configuration task.

The voucher service association file (*BRM_Home***/sys/data/config/pin_device_permit_map_voucher**) specifies the services (**/service** objects) with which voucher devices (**/device/voucher** objects) can be associated in your BRM system. By default, vouchers can be associated with these services:

- **/service/telco/gsm/data**

- **/service/telco/gsm/fax**
- **/service/telco/gsm/sms**
- **/service/telco/gsm/telephony**

> **Note:** To enable vouchers to be associated with accounts, the **pin_device_permit_map_voucher** file also includes the **/account** object in the list of services with which **/device/voucher** objects can be associated.

To use the voucher management features, you must load the voucher service association file into your BRM database. Before loading the file, you can customize the voucher service associations that it contains. See "Defining Device-to-Service Associations" in *BRM Developer's Guide*.

> **Note:** Voucher service associations are brand-aware and are associated with the root brand by default. See "Device Management and Brands" in *BRM Developer's Guide*.

Loading Voucher Details

> **Important:** This is a mandatory configuration task.

Voucher details are defined in the **pin_voucher_config** file. You run the **load_pin_voucher_config** utility to load the data into the **/config/voucher** object. See "About Voucher Details".

> **Important:** The **load_pin_voucher_config** 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*.

1. Edit the **pin_voucher_config** file **in** *BRM_Home***/sys/data/config.**
2. Save and close the **pin_voucher_config** file.
3. Use the following command to load the **pin_voucher_config** file:

```
load_pin_voucher_config voucher_input_file
```

For more information, see "load_pin_voucher_config".

Creating Vouchers for Testing

You need voucher devices in your database so you can create accounts and then create a set of vouchers.

> **Note:** This procedure creates an order with five vouchers.

To create a set of vouchers for testing:

1. In Voucher Administration Center, click the **New Order** icon or choose **File** - **New Order**.
2. Create an order using the following entries:
   - **Customer Info tab**:

     Customer name: Telecom Services

     Contact name: John Brown

     Email address: jbrown@telecom.com

     Address: 100 Main St

     City: Fremont

     State/Province: CA

     Zip/Postal: 99999

     Country: USA
   - **Vendor Info tab**:

     Vendor name: Cell Corp

     Contact name: Mary John

     Email address: jmary@cellcorp.com

     Address: 200 Main St

     City: RedWood City

     State/Province: CA

     Zip/Postal: 12234

     Country: USA
   - **Order Details tab**:

     Recharge card: V02 Voucher

     Order number: 1.35667

     Note: You can select either **Recharge Card** or **Order number**, but not both.

     Batch part number: BTH001

     Package part number: PT02

     Start serial number: 734536

     This is the voucher card starting number for this order, which you enter in the last box of the **Start serial number** box.

     Dealer: D001Airtel

     Total Vouchers: 2500

     If you select **Total Vouchers** and enter the quantity, the batch quantity is displayed in the **Number of batches** field.

     Number of batches: 1

If you select **Number of batches** and enter the batch quantity, the package quantity is displayed in the **Total Vouchers** box.

Days: 10

Date: 2/19/04

Note: You can select either **Days** or **Date**, but not both.

Deal Selected: Voucher_$300

3. Click **Generate File** to generate the request file.
4. Select or create a directory in which to generate the request files. The file name is **XXXXX.inp**. This file will be request file sent to the voucher card vendor.
5. Save the file as **XXXXX.out**. This file will be the vendor response file.
6. Process the vendor response file.
7. Load the response file. For complete instructions on using Voucher Administration Center, see Voucher Administration Center Help.

Uninstalling Voucher Manager

To uninstall Voucher Manager, run the *BRM_Home*/**uninstaller/VoucherMgr/uninstaller.bin**.

41

Voucher Manager Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) Voucher Manager utilities.

load_pin_dealers

Use this utility to load dealer information into the Oracle Communications Billing and Revenue Management (BRM) database. This utility loads dealer information from the **pin_dealer** file to a **/config/dealers** object.

You use this utility to configure a set of dealers for vouchers and identify the dealers associated with any specific card. The dealer names and codes are loaded into the database. See "About Dealers" and "Loading Dealer Details".

Note: You cannot load separate **/config/dealers** objects for each brand. All brands use the same object.

Caution: When you run the **load_pin_dealers** utility, it overwrites the existing dealers. If you are updating dealers, you cannot load new dealers only. You must load complete sets of dealers each time you run the **load_pin_dealers** utility.

Note: To connect to the BRM database, the **load_pin_dealers** utility requires 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_dealers [-v] [-d] dealer_input_file
```

Parameters

-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_dealers *any_other_parameter* **–v >** *filename*.**log**

-d
Writes error information for debugging purposes to the utility log file.

dealer_input_file
The name and location of the file that contains the dealer information. For example, the default file for dealer information is **pin_dealers**.

A sample definition file is provided in *BRM_Home*/**sys/data/config/pin_dealers**.

Use this syntax for entries:

```
dealer_name : dealer_code
```

Results

Reports success or displays an error.

By default, the file is located in the same directory as the utility and is called **default.pinlog**. You can specify a different name and location in the **Infranet.properties** file.

load_pin_recharge_card_type

Use this utility to load voucher information into the Oracle Communications Billing and Revenue Management (BRM) database. This utility loads voucher information such as dealer name, dealer code, recharge card type, and recharge card code from the **pin_recharge_card_type** file to a **/config/recharge_card_type** object. See "About Recharge Card Types" and "Loading Recharge Card Details".

This utility associates card types with their dealers.

Note: You cannot load separate **/config/recharge_card_type** objects for each brand. All brands use the same object.

Caution: When you run the **load_pin_recharge_card_type** utility, it overwrites the existing recharge card types. If you are updating a set of recharge card types, you cannot load new recharge card types only. You must load complete sets of recharge card types each time you run the **load_pin_recharge_card_type** utility.

Important: To connect to the BRM database, the **load_pin_recharge_card_type** utility requires 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_recharge_card_type [-v] [-d] card_type_input_file
```

Parameters

-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_recharge_card_type *any_other_parameter* **–v >** *filename*.**log**

-d
Writes error information for debugging purposes to the utility log file.

card_type_input_file
The name and location of the file that contains the recharge card type information. For example, the default file for recharge card type information is **pin_recharge_card_type**.

A sample definition file is provided in *BRM_Home*/**sys/data/config/pin_recharge_card_type**.

Use this syntax for entries:

```
recharge_card_type : recharge_card_code : dealer_name : dealer_code
```

Results

Reports success or displays an error.

By default, the file is located in the same directory as the utility and is called **default.pinlog**. You can specify a different name and location in the **Infranet.properties** file.

load_pin_voucher_config

Use this utility to load voucher information into the Oracle Communications Billing and Revenue Management (BRM) database. This utility loads voucher specific information such as batch part no, pack part number, pack quantity and batch quantity from the **pin_voucher_config** file to a **/config/voucher** object. See "About Voucher Details" and "Loading Voucher Details".

Note: You cannot load separate **/config/voucher** objects for each brand. All brands use the same object.

Caution: When you run the **load_pin_voucher_config** utility, it overwrites the existing dealers. If you are updating dealers, you cannot load new dealers only. You must load complete sets of dealers each time you run the **load_pin_voucher_config** utility.

Important: To connect to the BRM database, the **load_pin_voucher_config** utility requires 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_voucher_config [-d] [-v] voucher_input_file
```

Parameters

-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_vocher_config *any_other_parameter* **–v >** *filename*.**log**

-d
Writes error information for debugging purposes to the utility log file.

voucher_input_file
The name and location of the file that contains the voucher information. For example, the default file for voucher information is **pin_voucher_config**.

A sample definition file is provided in *BRM_Home*/**sys/data/config/pin_voucher_config**.

Use this syntax for entries:

```
batch_part_no : pack_part_no : pack_quantity : batch_quantity
```

Results

Reports success or displays an error.

By default, the file is located in the same directory as the utility and is called **default.pinlog**. You can specify a different name and location in the **Infranet.properties** file.

pin_voucher_expiration

Use this utility to move Oracle Communications Billing and Revenue Management (BRM) vouchers from the New state to the Expired state. See "Managing Expired Vouchers".

> **Note:** To connect to the BRM database, the **pin_voucher_expiration** utility requires 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

```
pin_voucher_expiration [-v] [-d]
```

Parameters

-v
Displays information about successful or failed processing as the utility runs.

> **Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-v** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace *filename*.**log** with the name of the log file:
>
> **pin_voucher_expiration** *any_other_parameter* **–v >** *filename*.**log**

-d
Writes error information for debugging purposes to the utility log file.

Results

Reports success or displays an error.

By default, the file is located in the same directory as the utility and is called **default.pinlog**. You can specify a different name and location in the **Infranet.properties** file.

Part X

Managing IP Address and APN Inventories

Part X describes how to manage IP address and APN inventories in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

- About IP Address Manager
- Installing and Configuring IP Address Manager and IP Address Administration Center
- Using the IP Address Manager APIs

42

About IP Address Manager

This chapter provides an overview of how you use Oracle Communications Billing and Revenue Management (BRM) IP Address Manager to manage your inventories of Internet Protocol (IP) addresses and Access Point Names (APNs). IP Address Manager includes the IP Address Administration Center GUI tool, which provides a graphical interface for managing these devices.

Important: IP Address Manager and IP Address Administration Center are optional components, not part of base BRM.

Before setting up IP Address Manager, you need to know the following information:

- Basic BRM concepts. See *BRM Concepts*.
- BRM system architecture. See "BRM System Architecture" in *BRM Concepts*.
- Basic information about how BRM manages devices. See "Managing Devices with BRM" in *BRM Developer's Guide*.
- How to create and edit BRM configuration files. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

About Managing IP Address and APN Device Inventories

BRM stores IP addresses and APNs as devices in the BRM database, so you can use them with the BRM Device Management framework. These devices are stored as BRM objects that can be associated with each other, as well as services and accounts. These IP addresses and APN devices are brand-aware and use device states to control their life cycles. You can use the default states and state restrictions, or create new ones to satisfy your business requirements.

You use these IP Manager components to manage IP address and APN devices:

- *IP Address Administration Center* is a GUI application used by operations personnel to manage IP address and APN inventories. For details, see the IP Address Administration Center online Help.
- IP Address Manager also includes command-line interfaces to manage your IP address and APN inventories, including creating and changing IP and APN devices, and associating them with accounts or services. For details, see "Using the IP Address Manager APIs".

About IP Address Manager

IP Address Manager stores IP addresses and APNs as device objects in your BRM database:

- **/device/ip** objects store information for a specific IP address, including the IP address number, its state, and associations to other objects.

  You can extend this object to suit your business needs. For details, see "Extending the IP Address Manager Storable Classes".

- **/device/apn** objects store information for specific APN devices.

  Note: The **/device/apn** subclass of the **/device** object does not contain any additional fields. The fields in **/device** are sufficient to use for APN addresses.

  This object includes the APN name and ID, its state, and associations to other objects.

Using the IP Address Manager APIs you can associate an IP address or APN with a customer account or any number of services. For details, see "Associating an IP Address with Accounts or Services".

About IP and APN Device States

IP address and APN devices are controlled by a device states.

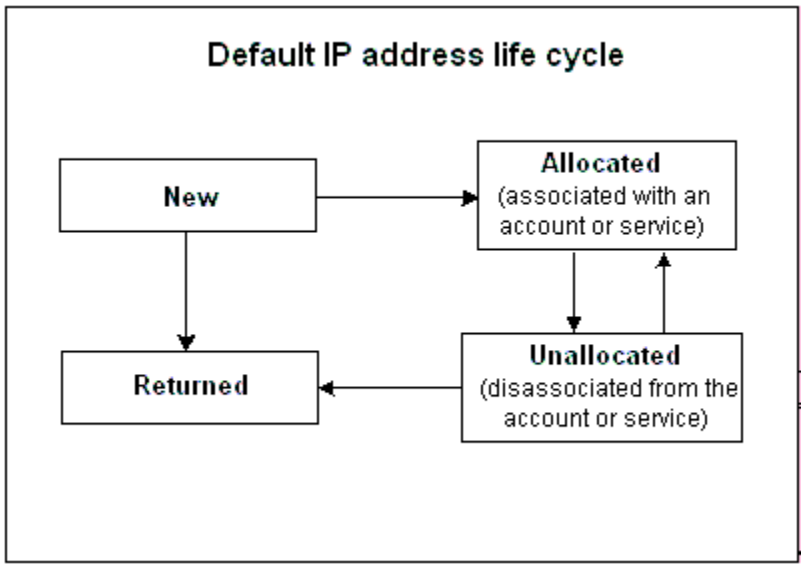
IP Address Device States

By default, the state of an IP device controls whether you can do the following:

- Associate or disassociate an IP address with a customer account or service.
- Delete an IP address.

Figure 42–1 shows the default IP address states and their relationships.

Figure 42–1 Default IP Address States and Relationships



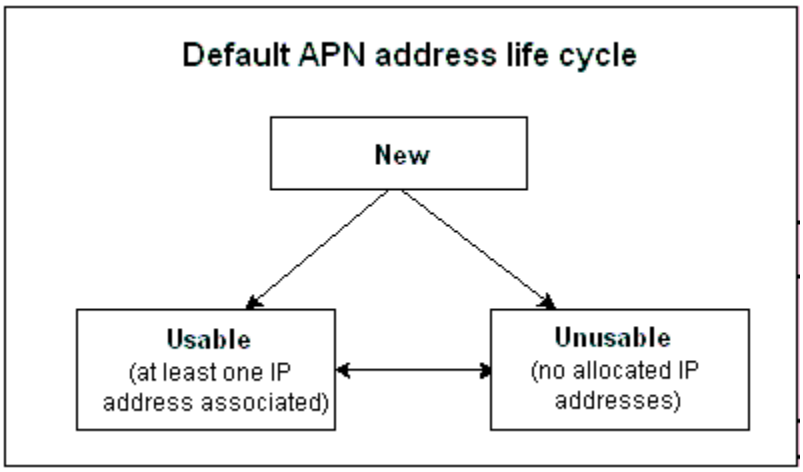
You can add additional states and customize their relationships to suit your business requirements. For details, see "Customizing IP Address and APN Device Life Cycles".

APN Device States

By default the state of an APN device controls whether the APN is usable or unusable.

Figure 42–2 shows the APN device states and their relationships. These changes are made automatically by IP Address Manager.

Figure 42–2 APN Device States and Relationships



You can add additional states and customize their relationships to suit your business requirements. For details, see "Customizing IP Address and APN Device Life Cycles".

Associated Services Lists

You can associate services with IP addresses and APNs to satisfy your business requirements. For details, see "Customizing IP Address and APN Service Association Lists".

IP Address Manager Configuration

Configuring IP Address Manager involves setting or changing IP address and APN life cycles.

To change IP address life cycles:

1. Set IP address states and give them name values in the **pin_device_state_ip** file.
2. Map IP address state name values to strings in the **ip_device_states**.*locale* file.
3. Create the list of services (as objects) that are permitted to associate with IP addresses in the **pin_device_permit_map_ip** file.

For more information, see "Installing and Configuring IP Address Manager and IP Address Administration Center".

To change APN life cycles:

1. Set APN states and give them name values in the **pin_device_state_apn** file.
2. Map APN state name values to strings in the **apn_device_states**.*locale* file.

3. Create the list of services (as objects) that are permitted to associate with APNs in the **pin_device_permit_map_apn** file.

For more information, see "Installing and Configuring IP Address Manager and IP Address Administration Center".

Important /device Object Fields

Table 42–1 lists the most important **/device** object fields used by IP Address Manager API.

Table 42–1 Important /device Object Fields

/device Object Field	IP Address Device Use	APN Device Use
PIN_FLD_DEVICE_ID	Stores the IP address (mandatory).	Stores the APN name (mandatory).
PIN_FLD_DESCR	Short description of the IP device.	Short description of the APN device.
PIN_FLD_STATE	Stores the IP device state.	Stores the APN device state.
PIN_FLD_ACCOUNT_OBJ	Stores the account used to create the IP device.	Stores the account used to create the APN device.

43

Installing and Configuring IP Address Manager and IP Address Administration Center

This chapter describes how to install Oracle Communications Billing and Revenue Management (BRM) IP Address Manager and IP Address Administration Center and how to configure and customize IP Address Manager.

For an overview of IP Address Manager and IP Address Administration Center, see "About IP Address Manager".

System Requirements

IP Address Manager is supported on the HP-UX IA64, Solaris, Linux and AIX operating systems. For information on disk space requirements for these operating systems, see "Disk Space Requirements" in *BRM Installation Guide*.

IP Address Administration Center is supported on the Windows platform and requires approximately 100 MB of disk space to download, extract, and install the software.

> **Note:** IP Address Administration Center requires the Java Runtime Environment (JRE). It is included in the IP Address Administration Center package and is approximately 50 MB. If the JRE was already installed with another BRM client application, it will not be reinstalled.

Software Requirements

Before installing IP Address Manager, you must install:

- Third-Party software, which includes the PERL 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 or Oracle 11g.

Before installing IP Address Administration Center, you must install:

- BRM. See "Putting Together Your BRM System" in *BRM Installation Guide*.
- **(Windows only)** An application such as WinZip for extracting compressed files.

Installing IP Address 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 IP Address 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 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.

   > **Caution:** 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 the temp_dir directory and enter this command:

   ```
   7.5.0_IPAddressMgr_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 IP Address Manager is **opt/portal/7.5**.

> **Note:** The installation program does not prompt you for the installation directory if BRM or **IP Address Manager** is already installed on the machine and automatically installs the package at the *BRM_Home* location.

5. Go to the directory where you installed the IP Address 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 IP Address Manager installation is now complete.

> **Important:** You should verify that the IP Address Manager Facilities Modules (FMs) were added to the Connection Manager (CM) configuration file.

Installing IP Address Administration Center

To install IP Address Administration Center on Windows:

1. Download the software to a temporary directory *(temp_dir)*.
2. Extract the downloaded **.zip** file to a temporary directory (*temp_dir*).
3. Go to *temp_dir* and run the **setup.exe** program. The installation wizard for IP Address Administration Center starts.
4. Answer the prompts in the installation wizard screens.

Your IP Address Administration Center installation is now complete.

Configuring IP Address Manager

After installing the IP Address Manager software, you will probably want to configure it to meet your business needs. This section explains the tasks necessary to configure IP Address Manager.

> **Note:** You can perform these tasks in any order, but you must install IP Address Manager first.

- You can define and use new IP address and APN states. This requires two procedures: defining and loading the state transition model, and loading the localized state names. See "Customizing APN Device States".

- You can define new service and account associations for IP address and APNs. See "Customizing IP Address and APN Service Association Lists".

Customizing IP Address and APN Device Life Cycles

This section explains how to control device life cycles by changing the default IP address and APN device states to meet your business needs. For a list of the default states and their relationships, see "About IP and APN Device States". You can change the state names and their relationships to each other. This section explains how to create new IP address and APN device states and state relationships. For information on changing the IP Address Manager opcodes to set these new states, see "Using the IP Address Manager APIs".

You can create brand-specific device states. See "Device Management and Brands" in *BRM Developer's Guide*. However, you need to be careful to create device state mappings that work if you change the brand of an APN. For example, if a device is in state 2 in one brand, it is still in that state when you change brands, but state 2 might be defined differently in the new brand.

Customizing IP Address Device States

Customizing IP address states includes adding or changing the device states, their names, and their relationships. These tasks are explained in the sections below.

Adding or Changing IP Address States

To customize device states, edit the **pin_device_state_ip** file in *BRM_Home*/**sys/data/config**, and then load it by running the **load_pin_device_state** utility. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

Adding or Changing Device State Names

When you customize device states, you need to add the new device state name to the list of device states that is displayed in IP Address Administration Center. To do so, you edit the **ip_device_states.en_US** sample file in the *BRM_Home*/**sys/msgs/ipdevicestates** directory. 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 this command:

```
load_localized_strings ip_device_states.locale
```

> **Note:** 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*.

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

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

Customizing APN Device States

Customizing an APN device state includes adding or changing the device states, their names, and their relationships. These tasks are explained in the sections below.

Adding or Changing Device States

To add or change device states, edit the **pin_device_state_apn** file in *BRM_Home*/**sys/data/config**, and then load it by running the **load_pin_device_state** utility. See "Defining the Device Life Cycle" in *BRM Developer's Guide*.

Adding or Changing Device State Names

When you customize device states, you need to add the new device state names to the list displayed in IP Address Administration Center. To do so, you edit the **apn_device_states.en_US** sample file in the *BRM_Home*/**sys/msgs/apndevicestates** directory. 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*.

This is the syntax:

```
load_localized_strings apn_device_states.locale
```

> **Note:** 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*.

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

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

Customizing IP Address and APN Service Association Lists

The following sections explain how to customize the lists of acceptable service and account associations.

Customizing the IP Service Association List

IP Address Manager allows you to associate services with an IP address. To do so, edit the *BRM_Home*/**sys/data/config/pin_device_permit_map_ip** file and then load it into the database by running the *BRM_Home*/**bin/load_pin_device_permit_map** utility.

For more information, see "Defining Device-to-Service Associations" and "load_pin_device_permit_map" in *BRM Developer's Guide*.

Customizing the APN Service Association List

You can customize which services are allowed to be associated with IP address devices. To do so, edit the *BRM_Home*/**sys/data/config/pin_device_permit_map_apn** file and then load it into the database by running the *BRM_Home*/**bin/load_pin_device_permit_map** utility.

For more information, see "Defining Device-to-Service Associations" and **load_pin_device_permit_map** in *BRM Developer's Guide*.

Troubleshooting

The following information will help you use IP Address Manager.

APN States Not Displayed in IP Address Administration Center

If an APN has no state displayed in IP Address Administration Center, the device states for the APN's brand have not been loaded into the BRM database. The brand administrator for the brand in question must do the following:

1. Load the **pin_device_state_apn** file for that brand.
2. Load the **apn_device_states.***locale* file if it changed or was not previously loaded.
3. Restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
4. Restart the IP Address Administration Center.

IP Address States Not Displayed in IP Address Administration Center

If an IP address has no state displayed in IP Address Administration Center, it means that the device states for the IP address's brand has not been loaded. The brand administrator for the brand in question must do the following:

1. Load the **pin_device_state_ip** file for that brand.
2. Load the **ip_device_states.***locale* file if it changed or was not previously loaded.
3. Restart the Connection Manager (CM). See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.
4. Restart the IP Address Administration Center.

Uninstalling IP Address Manager

To uninstall IP Address Manager, run the *BRM_Home*/**uninstaller/IPAddressMgr/uninstaller.bin**.

44

Using the IP Address Manager APIs

This chapter describes the tasks you can perform using the Oracle Communications Billing and Revenue Management (BRM) IP Address Manager APIs. Before performing any of these tasks, read this information:

- For an overview of IP Address Manager features, see "About IP Address Manager".
- Most IP Address Manager customizations will include modifying the IP Facilities Module (FM). For information and advice on how to do this, see "Writing a Custom Facilities Module" in *BRM Developer's Guide*.
- If you need to create subclasses of the default IP Address Manager storable classes, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

Using the IP Address Manager APIs

To create APN and IP address devices in the BRM database, the IP Address Manager opcodes do the following tasks:

- Creating a Single IP Address Device
- Creating a Range of IP Address Devices
- Customizing IP Address Creation
- Associating an IP Address with Accounts or Services
- Disassociating an IP Address Device from Accounts or Services
- Changing IP Device States from Unallocated to Returned
- Modifying an IP Address Device
- Setting the Brand on an IP Device
- Sorting IP Devices by Using Canonical IP Address
- Deleting an IP Address Device
- Creating an APN Device
- Associating APN with an Account or Service
- Modifying an APN Device
- Changing the APN Device State
- Setting the Brand for an APN Device
- Deleting an APN Device

Managing Your IP Address Device Life Cycle

This section explains how to use the IP Facilities Module (FM) and IP Policy FM to manage your IP address inventory.

Creating a Single IP Address Device

You create a single IP address device by calling the PCM_OP_IP_DEVICE_CREATE opcode and passing it an flist that includes:

- A type-only POID that specifies the device type of **/device/ip** in the PIN_FLD_POID field
- A valid IP address in the PIN_FLD_START_ADDRESS field
- The database being used

You add any validation checks or other business logic related to creating a single IP address device to PCM_OP_IP_POL_DEVICE_CREATE.

This is the calling sequence:

1. PCM_OP_IP_DEVICE_CREATE performs these tasks:
   - Verifies that the device type is **/device/ip**
   - Calls PCM_OP_DEVICE_SET_BRAND to get the correct brand of the IP address
   - Returns an error if the IP address already exists
   - Calls PCM_OP_DEVICE_CREATE
2. PCM_OP_DEVICE_CREATE performs this task:
   - Calls PCM_OP_DEVICE_POL_CREATE
3. PCM_OP_DEVICE_POL_CREATE performs this task:
   - Passes everything to PCM_OP_IP_POL_DEVICE_CREATE
4. PCM_OP_IP_POL_DEVICE_CREATE performs these tasks:
   - Determines the IP address number
   - Validates that the new IP/APN device combination is unique in the database.
   - Executes any other validation checks or business logic that you have added.
   - Returns an error if the IP device/APN device combination is a duplicate
   - Passes this information back to PCM_OP_DEVICE_CREATE
5. PCM_OP_DEVICE_CREATE performs these tasks:
   - Creates the object
   - Returns an error if the address is invalid

For more information on using PCM_OP_DEVICE_CREATE and PCM_OP_DEVICE_POL_CREATE, see "Creating /device Objects" in *BRM Developer's Guide*.

Creating a Range of IP Address Devices

You create a range of IP addresses the same way you create a single IP address, with a single call to PCM_OP_IP_DEVICE_CREATE. To create a range of addresses you enter valid start and end IP addresses in the PIN_FLD_START_ADDRESS and PIN_FLD_

END_ADDRESS fields. This opcode creates a range of **/device/ip** objects, one for each of the addresses, including the start and end addresses.

PCM_OP_IP_DEVICE_CREATE returns an error if the range is invalid.

Creating a Range of IP Addresses with a Subnet Mask

You create a range of all possible IP addresses under a specific subnet mask the same way you create a single IP address, with a single call to PCM_OP_IP_DEVICE_CREATE. To create an IP address/subnet mask range, you pass in the following information:

- A valid address in the PIN_FLD_START_ADDRESS field
- A valid subnet mask address in the PIN_FLD_SUBNET_MASK field

PCM_OP_IP_DEVICE_CREATE verifies that the range is valid and then creates all possible IP addresses in the range including the start and end addresses.

PCM_OP_IP_DEVICE_CREATE returns an error if the range is invalid.

Customizing IP Address Creation

Use the PCM_OP_IP_POL_DEVICE_CREATE policy opcode to customize IP address device creation. By default this opcode only checks to make sure the IP addresses to be created are not duplicates of existing IP addresses. This opcode is a hook designed for you to add any more validation checks or business logic that your business requires.

Associating an IP Address with Accounts or Services

You associate an IP address with any number of accounts or services by calling the PCM_OP_DEVICE_ASSOCIATE opcode. You also make device state changes to or from the **allocated** state with a call to PCM_OP_DEVICE_ASSOCIATE.

PCM_OP_DEVICE_ASSOCIATE requires the this information:

- An array of IP address objects to associate
- A PIN_FLD_SERVICES array containing an **/account** or **/service** POIDs in the PIN_FLD_ACCOUNT_OBJ or PIN_FLD_SERVICE_OBJ field
- The PIN_FLD_FLAGS field set to **0** (indicating an association)

This is the calling sequence:

1. PCM_OP_DEVICE_ASSOCIATE performs this task:
   - Calls PCM_OP_DEVICE_POL_ASSOCIATE
2. PCM_OP_DEVICE_POL_ASSOCIATE performs these tasks:
   - Executes any validation checks or other business logic that you added
   - Calls PCM_OP_IP_POL_DEVICE_ASSOCIATE if the device type is **/device/ip**
3. PCM_OP_IP_POL_DEVICE_ASSOCIATE performs these tasks:
   - Calls PCM_OP_DEVICE_SET_STATE to change the device state from **new** or **unallocated** to **allocated**
   - Executes any validation checks or business logic that you added
   - Calls PCM_OP_DEVICE_SET_STATE
4. PCM_OP_DEVICE_SET_STATE performs this task:

- Calls PCM_OP_IP_POL_DEVICE_SET_STATE

5. PCM_OP_DEVICE_POL_SET_STATE performs the following tasks:
   - Executes any other validation checks or business logic that you have added
   - Calls PCM_OP_IP_POL_DEVICE_SET_STATE to change the device state
6. PCM_OP_IP_POL_DEVICE_SET_STATE performs these tasks:
   - Executes any validation checks or business logic that you added
   - Returns the flist to PCM_OP_DEVICE_ASSOCIATE
   - Returns an error if an IP address is already allocated, or an IP address/service combination is not permitted
7. PCM_OP_DEVICE_ASSOCIATE performs this task:
   - Modifies the **/device/ip** object by adding the accounts or services passed to it

For more information on device associations using PCM_OP_DEVICE_ASSOCIATE and PCM_OP_DEVICE_POL_ASSOCIATE, see "Associating /service and /device Objects" in *BRM Developer's Guide*.

Disassociating an IP Address Device from Accounts or Services

The process of disassociating an IP address from accounts or services is much like associating IP addresses and accounts or services, except that you send in the input flist with PIN_FLD_FLAGS set to **1**. In this case the PCM_OP_DEVICE_POL_ASSOCIATE policy opcode:

- Confirms that none of the IP devices are associated with a service or account
- Changes the states from **allocated** to **unallocated**
- Disassociates the accounts or services by removing them from the **/device/ip** object

Changing IP Device States from Unallocated to Returned

You change IP device states between **unallocated** to **returned** using a call to PCM_OP_IP_DEVICE_SET_STATE. You pass in an array of the IP devices that get the state change, and the new state.

> **Note:** To make any device state changes to or from the **allocated** state, see Associating an IP Address with Accounts or Services.

You add any validation checks or other business logic related to changing the state of all instances of an IP device to PCM_OP_IP_POL_DEVICE_SET_STATE.

First you start by passing in an input flist to PCM_OP_IP_DEVICE_SET_STATE with this information:

- An array of POIDS to change (probably search results)
- The new state of for the devices

This is the calling sequence:

1. PCM_OP_IP_DEVICE_SET_STATE performs this task:
   - Calls PCM_OP_DEVICE_SET_STATE
2. PCM_OP_DEVICE_SET_STATE performs this task:

- Calls PCM_OP_DEVICE_POL_SET_STATE

3. PCM_OP_DEVICE_POL_SET_STATE performs these tasks:
   - Executes any other validation checks or business logic that you have added
   - Calls PCM_OP_IP_POL_DEVICE_SET_STATE if the device passed in is type **/device/ip**
4. PCM_OP_IP_POL_DEVICE_SET_STATE performs these tasks:
   - Executes any other validation checks or business logic that you have added
   - Returns the output flist to PCM_OP_DEVICE_POL_SET_STATE
5. PCM_OP_DEVICE_POL_SET_STATE performs this task:
   - Returns the output flist to PCM_OP_DEVICE_SET_STATE
6. PCM_OP_DEVICE_SET_STATE performs this task:
   - Makes the state change to all the **/device/ip** objects that have the same address

For more information on PCM_OP_DEVICE_SET_STATE and PCM_OP_DEVICE_POL_SET_STATE, see "Changing the State of a /device Object" in *BRM Developer's Guide*.

Modifying an IP Address Device

You use PCM_OP_IP_DEVICE_SET_ATTR to change the APN an IP is associated with. You call this opcode and pass it the POID of the **/device/ip** object and the change you want to make. You add any validation checks or other business logic related to modifying an IP device to PCM_OP_IP_POL_DEVICE_SET_ATTR.

This is the calling sequence:

1. PCM_OP_IP_DEVICE_SET_ATTR performs this task:
   - Calls PCM_OP_DEVICE_SET_ATTR if the device POID passed in is an **/device/ip** object
2. PCM_OP_DEVICE_SET_ATTR performs this task:
   - Calls PCM_OP_DEVICE_POL_SET_ATTR
3. PCM_OP_DEVICE_POL_SET_ATTR performs these tasks:
   - Performs any validation checks or business logic that you have added
   - Calls PCM_OP_IP_POL_DEVICE_SET_ATTR if the object passed in is type **/device/ip**
4. PCM_OP_IP_POL_DEVICE_SET_ATTR performs these tasks:
   - Validates that the object type is **/device/ip**.
   - Validates that the opcode being called is PCM_OP_IP_POL_DEVICE_SET_ATTR
   - Validates that the device state is not **allocated** or **returned**
   - Validates that the caller is not trying to change the IP device address
   - Executes any other validation checks or business logic that you have added
   - Returns an error if any of the validation checks fail
   - Returns the output flist to PCM_OP_DEVICE_POL_SET_ATTR

5. PCM_OP_DEVICE_POL_SET_ATTR performs this task:
   - Returns the flist to PCM_OP_DEVICE_SET_ATTR
6. PCM_OP_DEVICE_SET_ATTR performs this task:
   - Makes the changes to the **/device/ip** object

For more information on modifying devices using PCM_OP_DEVICE_SET_ATTR and PCM_OP_DEVICE_POL_SET_ATTR, see "Changing the Attributes of /device Objects" in *BRM Developer's Guide*.

Setting the Brand on an IP Device

You cannot change an IP device by itself. Instead, you use the instructions in Setting the Brand for an APN Device to change the brands of all IP devices associated with a single APN at the same time.

Sorting IP Devices by Using Canonical IP Address

The **/device/ip** object stores a canonical version of the device IP address that sorts more logically than a raw IP address. The PIN_FLD_DEVICE_CANONICAL_ID field contains an IP address that has been canonicalized by expanding all four parts to three digits each by using **0** (zero) as a placeholder. For example, the IP address 152.3.44.67 is stored in PIN_FLD_DEVICE_CANONICAL_ID as 152.003.044.067. Any standard ASCII search will sort these normalized IP address representations in simple numerical order.

Deleting an IP Address Device

To delete an IP address, call PCM_OP_IP_DEVICE_DELETE with a **/device/ip** object on the input flist. Before you delete IP devices, be sure to disassociate them from and accounts, services, or APNs. You add any validation checks or other business logic related to deleting an IP device to PCM_OP_IP_POL_DEVICE_DELETE.

Note: The IP Address Administrator does not support deleting devices. Using PCM_OP_DEVICE_DELETE is the only way to do this.

This is the calling sequence:

1. PCM_OP_IP_DEVICE_DELETE performs this task:
   - Calls PCM_OP_DEVICE_DELETE
2. PCM_OP_DEVICE_DELETE performs this task:
   - Calls PCM_OP_DEVICE_POL_DELETE
3. PCM_OP_DEVICE_POL_DELETE performs this task:
   - Calls PCM_OP_IP_POL_DEVICE_DELETE if the object passed in is type **/device/ip**
4. PCM_OP_IP_POL_DEVICE_DELETE performs these tasks:
   - Validates that the object type is **/device/ip**
   - Validates that the IP device is not in an allocated state
   - Executes any validation checks or other business logic that you have added
   - Returns an error if any of the validation checks fail

- Returns the flist to PCM_OP_DEVICE_POL_DELETE

5. PCM_OP_DEVICE_POL_DELETE performs this task:
   - Returns the flist to PCM_OP_DEVICE_DELETE
6. PCM_OP_DEVICE_DELETE performs this task:
   - Deletes the **/device/ip** object

Managing your APN Device Life Cycle

This section explains how to use the APN FM to manage your APN device inventory.

Creating an APN Device

To create an APN device, call PCM_OP_DEVICE_CREATE with a **/device/apn** object on the input flist. You add any validation checks or other business logic related to creating an APN device to PCM_OP_APN_POL_DEVICE_CREATE.

This is the calling sequence:

1. PCM_OP_DEVICE_CREATE performs this task:
   - Calls PCM_OP_DEVICE_POL_CREATE
2. PCM_OP_DEVICE_POL_CREATE performs this task:
   - If the device type is **/device/apn**, calls PCM_OP_APN_POL_DEVICE_CREATE
3. PCM_OP_APN_POL_DEVICE_CREATE performs these tasks:
   - Validates that the APN name is unique within the database passed in
   - Executes any other validation checks or business logic that you have added
   - Returns the output flist to PCM_OP_DEVICE_POL_CREATE
4. PCM_OP_DEVICE_POL_CREATE performs this task:
   - Returns the flist to PCM_OP_DEVICE_CREATE
5. PCM_OP_DEVICE_CREATE performs these tasks:
   - Creates the **/device/apn** object
   - Returns an error if the APN name is a duplicate

Associating APN with an Account or Service

You associate an APN with an account or service by calling the PCM_OP_DEVICE_ASSOCIATE opcode, passing it an flist with the following:

- The POID of the **/device/apn** object in the PIN_FLD_POID field
- An account or service POID in the PIN_FLD_ACCOUNT_OBJ or PIN_FLD_SERVICE_OBJ field

You add any validation checks or other business logic related to associating an APN with an account or service PCM_OP_APN_POL_DEVICE_ASSOCIATE.

This is the calling sequence:

1. PCM_OP_DEVICE_ASSOCIATE performs this task:
   - Calls PCM_OP_DEVICE_POL_ASSOCIATE

2. PCM_OP_DEVICE_POL_ASSOCIATE performs these tasks:
   - Executes any validation checks or business logic that you have added
   - Calls PCM_OP_APN_POL_DEVICE_ASSOCIATE
3. PCM_OP_APN_POL_DEVICE_ASSOCIATE performs these tasks:
   - Changes the device state from **new** to **allocated**
   - Executes any validation checks or business logic that you have added
   - Returns the flist to PCM_OP_DEVICE_POL_ASSOCIATE
4. PCM_OP_DEVICE_POL_ASSOCIATE performs this task:
   - Returns the flist to PCM_OP_DEVICE_ASSOCIATE
5. PCM_OP_DEVICE_ASSOCIATE performs this task:
   - Modifies the **/device/apn** object by adding the account or service passed to it

For more information on associating devices using PCM_OP_DEVICE_ASSOCIATE and PCM_OP_DEVICE_POL_ASSOCIATE, see "Associating /service and /device Objects" in *BRM Developer's Guide*.

Modifying an APN Device

To modify an APN, call the PCM_OP_DEVICE_SET_ATTR opcode, passing it a valid APN ID in the PIN_FLD_DEVICE_ID field. You add any validation checks or other business logic related to modifying an APN device to PCM_OP_APN_POL_DEVICE_SET_ATTR.

This is the calling sequence:

1. PCM_OP_DEVICE_SET_ATTR performs this task:
   - Calls PCM_OP_DEVICE_POL_SET_ATTR
2. PCM_OP_DEVICE_POL_SET_ATTR performs these tasks:
   - Executes any validation checks or business logic that you have added
   - Calls PCM_OP_APN_POL_DEVICE_SET_ATTR
3. PCM_OP_APN_POL_DEVICE_SET_ATTR performs these tasks:
   - Executes any validation checks or business logic that you have added
   - Validates that:

     The object type is **/device/apn**.

     The opcode being called is PCM_OP_APN_POL_DEVICE_SET_ATTR.

     None of the IP devices associated with the APN are in an **allocated** state.

     The caller is not trying to change the APN device address to one that is already in use.
   - Executes any other validation checks or business logic that you have added
   - Returns an error is any of the IP devices associated with the APN have a status of **allocated** or **returned**
   - Returns an error if you are trying to change the device ID and the new device ID is a duplicate of an existing device ID
   - Returns the output flist to PCM_OP_DEVICE_POL_SET_ATTR

4. PCM_OP_DEVICE_POL_SET_ATTR performs this task:
   - Returns the flist to PCM_OP_DEVICE_SET_ATTR
5. PCM_OP_DEVICE_SET_ATTR performs this task:
   - Makes the changes to the **/device/apn** object

For more information on modifying devices using PCM_OP_DEVICE_SET_ATTR and PCM_OP_DEVICE_POL_SET_ATTR, see "Changing the Attributes of /device Objects" in *BRM Developer's Guide*.

Changing the APN Device State

To change the state of an APN device, pass the **/device/apn** object POID and the new state to PCM_OP_DEVICE_SET_STATE. Add any validation checks or other business logic related to changing the state of an APN to PCM_OP_APN_POL_DEVICE_SET_STATE.

State change notes:

- The **new** device state is set by PCM_OP_APN_POL_DEVICE_CREATE only when an APN is created.
- Changing the device state from **new** to **usable** is allowed only as part of creating an IP device (which must be associated with an APN. In this case PCM_OP_APN_POL_DEVICE_SET_STATE confirms that PCM_OP_IP_POL_DEVICE_CREATE is part of the calling sequence. See Creating a Single IP Address Device and Creating a Range of IP Address Devices.

This is the calling sequence:

1. PCM_OP_DEVICE_SET_STATE performs this task:
   - Call PCM_OP_DEVICE_POL_SET_STATE
2. PCM_OP_DEVICE_POL_SET_STATE performs these tasks:
   - Executes any other validation checks or business logic that you have added
   - If the object passed in is the type **/device/apn**, calls PCM_OP_APN_POL_DEVICE_SET_STATE
3. PCM_OP_APN_POL_DEVICE_SET_STATE performs these tasks:
   - Executes any other validation checks or business logic that you have added
   - This opcode first confirms that none of the IP devices associated with the APN device are in an **allocated** state. If any are, it returns an error and rolls back the entire transaction
   - Returns the output flist to PCM_OP_DEVICE_POL_SET_STATE
   - If the device state change is from **new** to **usable**, confirms that is being done as a part of IP device creation by confirming that PCM_OP_IP_POL_DEVICE_CREATE is part of the calling sequence
4. PCM_OP_DEVICE_POL_SET_STATE performs this task:
   - Returns the output flist to PCM_OP_DEVICE_SET_STATE
5. PCM_OP_DEVICE_SET_STATE performs this task:
   - Makes the change to the **/device/apn** object

For more information on PCM_OP_DEVICE_SET_STATE and PCM_OP_APN_POL_DEVICE_SET_STATE, see "Changing the State of a /device Object" in *BRM Developer's Guide*.

Setting the Brand for an APN Device

To set the brand for an APN, call the PCM_OP_DEVICE_SET_BRAND opcode, passing it a valid brand POID in the PIN_FLD_POID field. You add any validation checks or other business logic related to setting the brand for an APN device to PCM_OP_APN_POL_DEVICE_SET_BRAND.

This is the calling sequence:

1. PCM_OP_DEVICE_SET_BRAND performs this task:
   - Calls PCM_OP_DEVICE_POL_SET_BRAND
2. PCM_OP_DEVICE_POL_SET_BRAND performs these tasks:
   - Executes any other validation checks or business logic that you have added
   - Calls PCM_OP_APN_POL_DEVICE_SET_BRAND
3. PCM_OP_APN_POL_DEVICE_SET_BRAND performs these tasks:
   - Validates that the device type is **/device/apn**
   - Validates that the opcode being called is PCM_OP_APN_POL_DEVICE_SET_BRAND
   - Validates that the APN has a state of **new**. or **usable**
   - Validates that none of the IP addresses associated with the APN are in an **allocated** state
   - Executes any other validation checks or business logic that you have added.
   - Returns an error message if any of the validation checks fail
   - Calls PCM_OP_DEVICE_SET_BRAND for each IP device associated with the APN, and passes it the **/device/apn** object
4. PCM_OP_DEVICE_SET_BRAND performs these tasks for each IP device it finds:
   - Calls PCM_OP_DEVICE_SET_BRAND and PCM_OP_DEVICE_POL_SET_BRAND
   - PCM_OP_DEVICE_POL_SET_BRAND calls PCM_OP_IP_POL_DEVICE_SET_BRAND
5. PCM_OP_IP_POL_DEVICE_SET_BRAND performs these tasks for each IP device:
   - Validates that the device type is **/device/ip**
   - Validates that the opcode being called is PCM_OP_IP_POL_DEVICE_SET_BRAND
   - Validates that the call is from PCM_OP_APN_POL_DEVICE_SET_BRAND
   - Validates that the state of the IP device is either **new** or **allocated**
   - Executes any other validation checks or business logic that you have added
   - If an error is returned while setting the brand on any of the IP devices, the entire transaction is aborted and rolled back (no IP devices get the name change)

- Returns the input flist or an error message to PCM_OP_DEVICE_POL_SET_BRAND

6. PCM_OP_DEVICE_POL_SET_BRAND performs this task:
   - Returns the input flist to PCM_OP_DEVICE_POL_SET_BRAND
7. PCM_OP_DEVICE_POL_SET_BRAND performs this task:
   - Returns the input flist to PCM_OP_DEVICE_SET_BRAND
8. PCM_OP_DEVICE_SET_BRAND performs this task for the APN device:
   - Sets the brand to the APN device object

For more information on setting brands using PCM_OP_DEVICE_SET_BRAND and PCM_OP_DEVICE_POL_SET_BRAND, see "Associating /devices and /brand Objects" in *BRM Developer's Guide*.

Deleting an APN Device

You delete an APN device by calling PCM_OP_DEVICE_DELETE with the object type of **/device/apn** in the input flist. Before you delete the APN device, be sure to disassociate any IP devices or services. You add any validation checks or other business logic related to deleting an APN device to PCM_OP_APN_POL_DEVICE_DELETE.

Note: The IP Address Administrator does not support deleting devices. Using PCM_OP_DEVICE_DELETE is the only way to do this.

This is the calling sequence:

1. PCM_OP_DEVICE_DELETE performs these tasks:
   - Calls PCM_OP_DEVICE_POL_DELETE
   - Executes any other validation checks or business logic that you have added
2. PCM_OP_DEVICE_POL_DELETE performs this task:
   - Calls PCM_OP_APN_POL_DEVICE_DELETE if the object passed to it is type **/device/apn**
3. PCM_OP_APN_POL_DEVICE_DELETE performs these tasks:
   - Validates that the object type is **/device/apn**
   - Probes for IP devices associated with the APN device; if it finds any, performs these tasks:
     - Confirms that none of the IP devices are allocated; if any are, returns an error
     - Deletes the IP devices by calling PCM_OP_DEVICE_POL_DELETE once for each device
     - PCM_OP_DEVICE_POL_DELETE performs this task:

       Returns the flist to PCM_OP_DEVICE_DELETE
     - PCM_OP_DEVICE_DELETE performs this task:

       Deletes **/device/ip** objects that were associated with the APN device
   - Executes any other validation checks or logic that you have added

- Returns an error if any of the validation checks fail
- Returns the flist to PCM_OP_DEVICE_POL_DELETE

4. PCM_OP_DEVICE_POL_DELETE performs this task:
   - Returns the flist to PCM_OP_DEVICE_DELETE
5. PCM_OP_DEVICE_DELETE performs this task:
   - Deletes the **/device/apn** object

Extending the IP Address Manager Storable Classes

You use the BRM Storable Class Editor to extend the **/device/ip** storable class to meet your business needs. For more information, see "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide*.

Adding Business Logic to the IP Address and APN Policy FMs

The IP and APN policy Facilities Modules (FMs) are designed for you to add your own customizations, such as additional validation checks. For information on extending these FMs, see "Adding and Modifying Policy Facilities Modules" in *BRM Developer's Guide*.

Part XI

Managing Dropped Calls and Continuation Calls

Part XI describes how to manage dropped calls in an Oracle Communications Billing and Revenue Management (BRM) system. It contains the following chapters:

- About Finding Dropped Calls and Continuation Calls
- Configuring Your System for Dropped Calls and Continuation Calls

45

About Finding Dropped Calls and Continuation Calls

This chapter describes how Oracle Communications Billing and Revenue Management (BRM) finds dropped calls and continuation calls.

Before reading this document, you should be familiar with the following concepts:

- BRM discounting. See "About Discounts" in *BRM Configuring Pipeline Rating and Discounting*.
- BRM prepaid authentication, authorization, and accounting (AAA). See "Understanding Prepaid AAA".

About Dropped Calls and Continuation Calls

A customer's phone call may terminate unexpectedly for a variety of technical reasons, such as the mobile phone moving out of the wireless network's range or network interference. The customer may then make another call to the same phone number to resume the connection.

You can use the BRM dropped calls feature to identify when a customer's call is dropped (called a *dropped call*) and then resumed again through a subsequent call (called a *continuation call*). This allows you to compensate the customer for any inconvenience by discounting the dropped call, discounting the continuation call, or granting a credit to the customer.

About the Criteria for Finding Dropped Calls

BRM identifies *dropped calls* by reading the termination cause applied by the network switch and specified in the event data record (EDR) or event. You specify which termination causes qualify as a dropped call by using the **pin_telco_aaa_params.xml** file (real-time rating) or the FCT_DroppedCall registry entries (batch rating).

For more information, see the following:

- For real-time rating, see "Specifying the Termination Causes for Dropped Calls".
- For batch rating, see "Configuring Batch Rating to Find Dropped Calls and Continuation Calls".

About the Criteria for Finding Continuation Calls

BRM identifies *continuation calls* by using the following criteria:

- **The call time**: Continuation calls must occur within a specified time frame after the dropped call. You can specify a maximum time interval for each service type that you support. If you do not specify a value, there is no time limit as long as both calls occur within the same billing cycle.
- **The call's placement after the dropped call:** Continuation calls must occur within a specified number of calls after the dropped call. For example, you can specify that BRM checks only the customer's first call after the dropped call. If you do not specify a value, BRM checks all calls made by the customer.
- **The called party number:** You can specify whether continuation calls must be to the same called number as the dropped call or if calls to other numbers can be continuation calls. If you do not specify a value, continuation calls must be to the same called number as the dropped call.

You set the above criteria by creating a service-level extended rating attribute (ERA). See "Creating the Dropped Calls ERA".

You can configure BRM to use additional criteria for finding continuation calls by rewriting a policy opcode (real-time rating) or by using the FCT_DroppedCall registry entries (batch rating). For more information, see the following:

- For real-time rating, see "Specifying the Rules for Finding Continuation Calls".
- For batch rating, see "Configuring Batch Rating to Find Dropped Calls and Continuation Calls".

How Batch Rating Detects Dropped Calls and Continuation Calls

During batch rating, the pipeline checks EDRs to see if they meet the criteria for a dropped call. When an EDR meets the criteria, the pipeline flags it as a dropped call and stores configuration information about the dropped call in internal pipeline memory. The pipeline then checks the caller's subsequent calls to determine whether they meet the criteria for a continuation call. When a call meeting the criteria is found, the pipeline adds to the EDR a continuation call flag and information about the dropped call, such as the dropped call's duration.

The batch rating process uses the FCT_DroppedCall pipeline module to detect and flag dropped calls and continuation calls. You use the module's registry entries to specify the following:

- The EDR field and values used to identify dropped calls.
- (Optional) The EDR fields used to identify continuation calls.
- (Optional) The dropped call EDR fields to add to continuation calls.
- General connection parameters, such as the name and location of the dropped calls data file.

You use FCT_DroppedCall in the rating pipeline, the rerating pipeline, and the recycling pipeline. See "FCT_DroppedCall" in *BRM Configuring Pipeline Rating and Discounting*.

How Batch Rating Identifies Dropped Calls

When a phone call ends, the network switch records the termination cause in the customer's call details record (CDR). The BRM input grammar file maps this termination cause to the DETAIL.CALL_COMPLETION_INDICATOR EDR field. You specify which EDR field values qualify as a dropped call by using the FCT_DroppedCall registry entries.

Note: FCT_DroppedCall uses the termination cause specified in the DETAIL.CALL_COMPLETION_INDICATOR EDR field by default. You can configure the module to use criteria from a different EDR field.

To identify dropped calls, the FCT_DroppedCall module performs these tasks:

1. Checks whether there is a valid dropped call service-level ERA associated with the service.
   - If there is an ERA, FCT_DroppedCall proceeds to the next step.
   - If there is not an ERA, the module sets the DETAIL.DROPPED_CALL_STATUS EDR field to **0**.
2. Determines whether the EDR meets the criteria specified in the registry file. If it meets the criteria, the module flags the call as a dropped call by setting the DETAIL.DROPPED_CALL_STATUS EDR field to **1**. It also writes to pipeline memory a list of configurable field values.
3. Writes the in-memory data to the dropped calls data file. See "About the Dropped Calls Data File".

How Batch Rating Identifies Continuation Calls

After identifying a dropped call, FCT_DroppedCall checks the customer's subsequent EDRs to determine if any of them meet the criteria for a continuation call.

When examining an EDR, the module determines whether it meets the criteria specified in the dropped call ERA and FCT_DroppedCall registry entries, and then performs the following:

- If the call meets the criteria, the module sets the EDR's DETAIL.DROPPED_CALL_STATUS field to **2** to indicate that it is a continuation call.
- If the call meets the criteria and the DETAIL.DROPPED_CALL_STATUS EDR field is already set to **1**, the module changes the EDR field to **3** to indicate that it is both a dropped call and a continuation call.
- If the call does not meet the criteria and exceeds either the maximum call time or the maximum number of intermediate calls, the module sets the EDR's DETAIL.DROPPED_CALL_STATUS field to **4** to indicate that it didn't meet the criteria for a dropped call or a continuation call.
- If the call does not meet the criteria, but does not exceed both the maximum call time and maximum number of intermediate calls, the module:
  - Increments a counter stored in memory that tracks the number of intermediate calls.
  - Records the call's starting timestamp.
  - Sets the EDR's DETAIL.DROPPED_CALL_STATUS field to **4** to indicate that it didn't meet the criteria for a dropped call or a continuation call.
  - Examines the customer's next call.

At the end of each transaction, the module writes the in-memory data to the dropped calls data file. See "About the Dropped Calls Data File".

About the Dropped Calls Data File

FCT_DroppedCall stores in internal pipeline memory information about a dropped call, including the EDR fields for identifying a continuation call and the list of EDR fields to attach to the continuation call EDR. When the module finds the matching continuation call, it closes the dropped call and removes information about the dropped call from memory.

At the end of each transaction, the module backs up the in-memory data to a dropped calls data file. This enables you to reload the data into memory if the system crashes or restarts.

FCT_DroppedCall updates the data file at the end of each transaction to match the information stored in memory, such as:

- Adding any new dropped calls written to memory.
- Adding the starting timestamp for the last processed intermediate call.
- Updating a dropped call's counter of intermediate calls.
- Deleting any dropped calls that were removed from memory.

You configure how often to purge old calls from memory and the data file by using the **RemoveLimit** semaphore file entry. See "FCT_DroppedCall" in *BRM Configuring Pipeline Rating and Discounting* and "Purging Old Call Data from Memory".

Note: Account Migration Manager (AMM) does not move the dropped calls data file between pipelines. If an account is migrated, you must manually move any associated dropped call entries to the new pipeline.

About Recycling Dropped Calls and Continuation Calls

The pipeline rejects some EDRs because they fail validation or because a module added an error code. This can prevent an EDR from being identified as a dropped call or a continuation call, depending on where in the pipeline the EDR is rejected.

Table 45–1 describes how rejected EDRs affect FCT_DroppedCall's ability to detect dropped calls and continuation calls.

Table 45–1 Rejected EDR Affect on FCT_DroppedCall

EDR Type	Rejected Before or After FCT_DroppedCall	Description
Dropped call EDR	Before	The EDR is not identified as a dropped call. Any subsequent calls are not identified as a continuation call.
Dropped call EDR	After	The EDR is identified as a dropped call and stored in memory. The module can still find the call's associated continuation call. During the recycling process, FCT_DroppedCall does not re-evaluate any EDR that was already processed successfully by the module. Therefore, the module does not re-evaluate the dropped call EDR.

Table 45–1 (Cont.) Rejected EDR Affect on FCT_DroppedCall

EDR Type	Rejected Before or After FCT_DroppedCall	Description
Continuation call EDR	Before	The EDR is not initially tagged as a continuation call. During the recycling process, the rejected EDR is reprocessed and can be identified as a continuation call. **Note:** The module flags the first EDR it processes that meets the criteria for a continuation call. Therefore, if multiple EDRs meet the criteria, the module may flag a different EDR as the continuation call.
Continuation call EDR	After	The EDR is tagged as a continuation call. During the recycling process, FCT_DroppedCall does not re-evaluate any EDR that was already processed successfully by the module. Therefore, the module does not re-evaluate the continuation call EDR.

About Batch Rerating and Dropped Calls

During the rerating process, the FCT_DroppedCall module re-evaluates each EDR to determine whether it qualifies as a dropped call or a continuation call, using the same method it did during the rating process.

FCT_DroppedCall can find and flag continuation calls during the rerating process only if both the dropped call EDR and the continuation call EDR are included in the rerating job. To ensure that both EDRs are included in the rerating job, configure your rerating trigger to check for continuation calls. If a continuation call is found, the trigger should backdate the start time to include the dropped call.

To support dropped calls and continuation calls during the rerating process, add the FCT_DroppedCall module to your batch rerating pipeline and real-time rerating pipeline.

For more information about rerating, see "About Rerating Pipeline-Rated Events" in *BRM Setting Up Pricing and Rating*.

How Real-Time Rating Detects Dropped Calls and Continuation Calls

During real-time rating, BRM stores information about ongoing prepaid sessions in **/active_session** objects. When the call ends, BRM records the termination cause and determines whether the call qualifies as a dropped call. When a call meets the criteria, BRM flags it as a dropped call.

When BRM authorizes or reauthorizes subsequent prepaid sessions, it determines whether the session qualifies as a continuation call by comparing the current session with the caller's previous call sessions. If the call meets the criteria, it is flagged as a continuation call.

BRM uses the PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL helper opcode and the PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL policy opcode to detect and flag continuation calls.

How Real-Time Rating Detects Dropped Calls

When a prepaid AAA call ends, real-time rating performs these tasks to find and tag dropped calls:

1. Records the call's termination cause in the **/active_session** object's PIN_FLD_TERMINATE_CAUSE field.
2. Determines whether the call qualifies as a dropped call by comparing the termination cause with the value specified in the **/config/aaa/gsm/***xxx* object's PIN_FLD_DROPPED_CALL_TERMINATE_CAUSE field:
   - If it qualifies as a dropped call, BRM flags the call as a dropped call by setting the **/active_session** object's PIN_FLD_CALL_TYPE field to **1**.
   - If it qualifies as a dropped call and the call is already flagged as a continuation call, BRM flags the call as both a dropped call and a continuation call by setting the **/active_session** object's PIN_FLD_CALL_TYPE field to **3**.
   - If it does not qualify as a dropped call, BRM does not modify the PIN_FLD_CALL_TYPE field.

How Real-Time Rating Detects Continuation Calls

Real-time rating performs these tasks during the prepaid AAA process to find and tag continuation calls:

1. The AAA opcode calls the PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL helper opcode at the TAG_SESSION processing stage.
2. PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL searches for the dropped call ERA associated with the service. If the ERA is not present, the opcode returns to the calling opcode.
3. PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL determines whether the call is already flagged as a continuation call:
   - If it is not flagged, the helper opcode continues to the next step.
   - If it is already flagged and the calling opcode is PCM_OP_TCF_AAA_STOP_ACCOUNTING, the helper opcode optionally deletes any redundant **/active_session** objects from memory and then returns to the calling opcode.
   - If it is already flagged and the calling opcode is any opcode other than PCM_OP_TCF_AAA_STOP_ACCOUNTING, the helper opcode returns to the calling opcode.
4. PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL searches through the existing **/active_session** objects in memory to find all **/active_session** objects with the same service type and caller number as the current call.
5. PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL sorts the **/active_session** objects that met the criteria by PIN_FLD_CREATED_T, in decreasing order.
6. PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL loops through the sorted **/active_session** objects to find ones that match the dropped call termination cause.
7. At the MATCH_CONTINUOUS_CALL processing stage, PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL sends the current call, the dropped call, the dropped call ERA, the **/config/aaa/gsm/***xxx* object, billing cycle information, and the list of intermediate **/active_session** objects to the policy opcode specified in the

/config/opcodemap/tcf object. By default, **/config/opcodemap/tcf** is configured to call PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL.

> **Note:** You can change which policy opcodes are called at the MATCH_CONTINUOUS_CALL processing stage by using the **pin_config_opcodemap_tcf** configuration file. See "Customizing the Criteria for Finding Continuation Calls".

8. PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL checks whether the current **/active_session** object meets the criteria specified in the dropped call ERA.
9. PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL returns to PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL the PIN_FLD_RESULT field set to one of the following:
   - **0** to indicate that the current call *is not* a continuation call.
   - **1** to indicate that the current call *is* a continuation call.
   - **2** to indicate that the current call *is not* a continuation call because it exceeds the maximum time duration or maximum number of intermediate calls.
10. PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL performs one of the following, depending on the value of the PIN_FLD_RESULT flist field:
    - If PIN_FLD_RESULT is set to **0**, the opcode flags the current call's **/active_session** object as a normal call by setting the PIN_FLD_CALL_TYPE field to **0**.
    - If PIN_FLD_RESULT is set to **1**, the opcode flags the current call's **/active_session** object as a continuation call by setting the PIN_FLD_CALL_TYPE field to **2**. It also adds the duration of the dropped call to the PIN_FLD_DROPPED_CALL_QUANTITY field, and the POID of the dropped call's **/active_session** object to the PIN_FLD_DROPPED_CALL_ASO_POID field.
    - If PIN_FLD_RESULT is set to **2**, the opcode flags the current call's **/active_session** object as a normal call by setting the PIN_FLD_CALL_TYPE field to **0**. It also stops iterating the **/active_session** objects.

About Storing Dropped Call Data during Real-Time Rating

When performing prepaid AAA, BRM stores information about on-going sessions in **/active_session** objects. When the prepaid session ends, BRM transfers the information to an **/event/session** object and either keeps or deletes the **/active_session** object, depending on how the **DeletedFlag** field is set in the **/config/aaa/gsm/***xxx* object.

When configured for dropped calls, BRM automatically keeps **/active_session** objects for any service type that supports the dropped calls feature. This allows BRM to search through previous call sessions. You specify which services support the dropped calls feature in the service-specific **/config/aaa/gsm/***xxx* object. See "Specifying the Termination Causes for Dropped Calls".

When a prepaid session ends, BRM deletes any redundant **/active_session** objects to reduce disk space. That is, BRM searches through the **/active_session** objects and deletes objects that meet all of these criteria:

- Have the same caller number as the current call.
- Have the same service type as the current call.

- Have any of the following:
  - A timestamp that exceeds the maximum duration specified in the dropped call ERA.
  - A call counter that surpasses the maximum number of intermediate calls specified in the dropped call ERA.
  - A billing cycle that is different than that of the current call.

If the current call is flagged as a continuation call, BRM also deletes the dropped call, the continuation call, and all intermediate calls that aren't an unidentified dropped call themselves.

About Real-Time Rerating and Dropped Calls

The real-time rerating process does not re-evaluate whether a call qualifies as a dropped call or a continuation call; instead, it relies upon the event's existing dropped call and continuation call flags.

About Applying Discounts and Credits to Dropped Calls and Continuation Calls

You can compensate customers for dropped calls by doing one of the following:

- Discounting the dropped call, based on the call termination value.
- Discounting the continuation call, based on the duration of the dropped call.
- Granting a credit to the customer that can be applied in the current billing cycle or the next billing cycle.

You specify how to compensate customers by using BRM discounting. During the discounting process, BRM determines whether the event is a dropped call or a continuation call and then applies the appropriate discount or credit.

For more information about BRM discounting, see "About Discounts" in *BRM Configuring Pipeline Rating and Discounting*.

46

Configuring Your System for Dropped Calls and Continuation Calls

This chapter explains how to configure your Oracle Communications Billing and Revenue Management (BRM) system to support dropped calls and continuation calls. For more information about dropped calls and continuation calls, see "About Finding Dropped Calls and Continuation Calls".

Setting Up Your System to Identify Dropped Calls and Continuation Calls

To set up your system to identify dropped calls and continuation calls, perform these tasks:

1. Creating the Dropped Calls ERA
2. Configuring Batch Rating to Find Dropped Calls and Continuation Calls
3. Configuring Real-Time Rating to Find Dropped Calls and Continuation Calls

Creating the Dropped Calls ERA

You create the dropped calls extended rating attribute (ERA) by editing the *BRM_Home*/**sys/data/config/pin_config_provisioning_tags_droppedcall.xml** file. You then load the file into the database's **/config/provisioning_tag** object by using the **load_config_provisioning_tags** utility.

The **pin_config_provisioning_tags_droppedcall.xml** file specifies the services to which the provisioning tag applies, the opcodes to run when a customer purchases or cancels a product or discount that contains the tag, and the fields to pass in the opcode's input flist.

For the dropped calls ERA, you pass the following fields in Table 46–1 to the PCM_OP_SUBSCRIPTION_PROVISION_ERA opcode:

Table 46–1 Dropped Calls ERA Fields

XML Field	Description
MAX_TIME_TO_ CONTINUATION_CALL	Specifies the allowable duration, in seconds, between a dropped call and a continuation call. The duration specifies the time between the end of a dropped call and the start of the continuation call. For example, if a dropped call ends at 10 a.m. and the MAX_TIME_TO_CONTINUATION_CALL is 300 (5 minutes), the continuation call starting timestamp must be between 10:00:00 and 10:04:59, inclusive. If you do not specify a value, there's no time limit between the dropped call and the continuation call, as long as they both occur within the same billing cycle.
MAX_INTERVENING_CALLS	Specifies the allowable number of intermediate calls between a dropped call and a continuation call. For example, if you specify **0**, BRM checks the customer's first call after the dropped call only. If you do not specify a value, BRM allows an infinite number of calls between the dropped call and the continuation call.
SAME_CALLED_PARTY	Specifies whether the continuation call must be to the same called number as the dropped call or if it can be to another number. **0** = The continuation call can be to any number. **1** = The continuation call must be to the same number as the dropped call. That is, it must have the same B number. If you do not specify a value, BRM requires the same called number.

If you do not specify any profile values, the dropped calls feature requires only that the continuation call be made to the same number as the dropped call and that the call occurs within the same billing cycle as the dropped call.

For more information on how to edit the **pin_config_provisioning_tags_ droppedcall.xml** file, see "Working with Provisioning Tags" in *BRM Setting Up Pricing and Rating*.

Important: The dropped call promotional description is included in the *BRM_Home***/sys/msgs/eradescr/era_descr.en_US** file. Customer Center cannot display the dropped call ERA until you reload the **era_ descr.en_US** file with the **load_localized_strings** utility. See "load_ localized_strings" in *BRM Developer's Guide*.

Sample pin_config_provisioning_tags_droppedcall.xml File

The following sample **pin_config_provisioning_tags_droppedcall.xml** file specifies that when a customer purchases a GSM telephony service, BRM calls the PCM_OP_ SUBSCRIPTION_PROVISION_ERA opcode with the following information in the input flist:

- The customer's account and service POIDs
- The service profile ERA name set to DROPPED_CALL
- MAX_TIME_TO_CONTINUATION_CALL set to 600 seconds
- MAX_INTERVENING_CALLS set to 10

- SAME_CALLED_PARTY set to 1

The opcode then creates a **/profile/serv_extrating** object that stores a service-level profile ERA named DROPPED_CALL. When the customer cancels the GSM telephony service, the opcode deletes the **/profile/serv_extrating** object.

```
<BusinessConfiguration xmlns="http://www.portal.com/schemas/BusinessConfig"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.portal.com/schemas/BusinessConfig
 business_configuration.xsd">
    <ProvisioningTagConfiguration>
      <ProvisioningTagList>
        <ProvisioningTag name="DroppedCall">
          <PermittedTypes>/service/telco/gsm/telephony</PermittedTypes>
          <OpcodeList>
            <OpcodeName>PCM_OP_SUBSCRIPTION_PROVISION_ERA</OpcodeName>
            <OpcodeNumber>9066</OpcodeNumber>
            <OpcodeMode>0</OpcodeMode>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                <OpcodeParamValue>0.0.0.0 /profile/serv_extrating -1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_FLAGS</OpcodeParamName>
                <OpcodeParamValue>0</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_ACCOUNT_OBJ</OpcodeParamName>
                <OpcodeParamValue>$ACCOUNT$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SERVICE_OBJ</OpcodeParamName>
                <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_NAME</OpcodeParamName>
                <OpcodeParamValue>DROPPED_CALL</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_EXTRATING.PIN_FLD_REFERENCE_
COUNT</OpcodeParamName>
                <OpcodeParamValue>1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_DATA_ARRAY[0].PIN_FLD_
NAME</OpcodeParamName>
                <OpcodeParamValue>MAX_INTERVENING_CALLS</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_DATA_ARRAY[0].PIN_FLD_
VALUE</OpcodeParamName>
                <OpcodeParamValue>10</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_DATA_ARRAY[1].PIN_FLD_
NAME</OpcodeParamName>
                <OpcodeParamValue>SAME_CALLED_PARTY</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_DATA_ARRAY[1].PIN_FLD_
VALUE</OpcodeParamName>
```

```
                <OpcodeParamValue>1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_DATA_ARRAY[2].PIN_FLD_
NAME</OpcodeParamName>
                <OpcodeParamValue>MAX_TIME_TO_CONTINUATION_CALL</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_INHERITED_INFO.PIN_FLD_DATA_ARRAY[2].PIN_FLD_
VALUE</OpcodeParamName>
                <OpcodeParamValue>600</OpcodeParamValue>
            </OpcodeParamsList>
        </OpcodeList>

        <OpcodeList>
            <OpcodeName>PCM_OP_SUBSCRIPTION_PROVISION_ERA</OpcodeName>
            <OpcodeNumber>9066</OpcodeNumber>
            <OpcodeMode>1</OpcodeMode>

            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_POID</OpcodeParamName>
                <OpcodeParamValue>0.0.0.0 /profile/serv_extrating -1</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_FLAGS</OpcodeParamName>
                <OpcodeParamValue>0</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_ACCOUNT_OBJ</OpcodeParamName>
                <OpcodeParamValue>$ACCOUNT$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_SERVICE_OBJ</OpcodeParamName>
                <OpcodeParamValue>$SERVICE$</OpcodeParamValue>
            </OpcodeParamsList>
            <OpcodeParamsList>
                <OpcodeParamName>PIN_FLD_NAME</OpcodeParamName>
                <OpcodeParamValue>DROPPED_CALL</OpcodeParamValue>
            </OpcodeParamsList>
        </OpcodeList>

    </ProvisioningTag>
    </ProvisioningTagList>
    </ProvisioningTagConfiguration>
    </BusinessConfiguration>
```

Configuring Batch Rating to Find Dropped Calls and Continuation Calls

To set up batch rating to find dropped calls and continuation calls, configure the FCT_DroppedCall module to run after the FCT_Account module. See "FCT_DroppedCall" in *BRM Configuring Pipeline Rating and Discounting*.

When you configure the FCT_DroppedCall module, you specify the following:

- The EDR field and value used to identify dropped calls. See "Specifying the EDR Fields for Finding Dropped Calls".
- The EDR fields and values used to identify continuation calls. See "Specifying the EDR Fields for Identifying Continuation Calls".

- How to enrich the continuation call EDR. See "Mapping Dropped Call Fields to Continuation Call Fields".

Specifying the EDR Fields for Finding Dropped Calls

You specify the EDR field for identifying dropped calls by using the **CheckField** registry section. This section lists the EDR field name and its permissible values.

> **Note:** Only one EDR field can be used to identify a dropped call.

The **CheckField** section uses the following format:

```
CheckField
{
  Name = EDR_field
  Value = Field_value
}
```

Use the **CheckField.Name** entry to specify the EDR field name, such as **DETAIL.CALL_COMPLETION_INDICATOR**, and use the **CheckField.Value** entry to specify the EDR value, such as **1**. If more than one value qualifies an EDR as a dropped call, enter multiple values separated by a comma (**,**) with no spaces; for example: **5,6,7**. BRM interprets the comma as a Boolean OR value.

Specifying the EDR Fields for Identifying Continuation Calls

By default, the FCT_DroppedCall module writes the following dropped call EDR fields to memory and uses them to identify a continuation call:

- DETAIL.A_NUMBER
- DETAIL.B_NUMBER

> **Note:** This field is used only if you specified that continuation calls must be to the same called number as the dropped call. See "About the Criteria for Finding Dropped Calls".

- DETAIL.CHARGING_END_TIMESTAMP
- DETAIL.CUST_A.BILL_NEXT_DATE

An EDR is flagged as a continuation call if its EDR field values match those of the dropped call EDR.

You can use additional EDR fields to identify continuation calls by using the **WrittenFields** registry section. You list the EDR fields to write to memory by using dummy key values, such as **1** and **2**, as shown below:

```
WrittenFields
{
  1 = EDR_field
  2 = EDR_field
  3 = EDR_field
}
```

Mapping Dropped Call Fields to Continuation Call Fields

By default, the FCT_DroppedCall module does not enrich the continuation call EDR. You can add information from the dropped call EDR to the continuation call EDR by

using the **AddedFields** registry section. This section maps dropped call EDR fields to continuation call EDR fields.

> **Important:** When you map a dropped call EDR field to a continuation call EDR field, both fields must have the same data type. You can find a field's data type by reading the container description file (**container.dsc**).

The **AddedFields** section uses the following format:

```
AddedFields
{
  Fieldx
  {
    ContinuationCallField = EDR_Field
    DroppedCallField = EDR_Field
  }
}
```

Each **Field**x section maps one dropped call EDR field to one continuation call EDR field. You create a **Field*x*** section for each pair of EDR fields that you want to map. For example, to map three EDR pairs, create a **Field1** section, a **Field2** section, and a **Field3** section.

Configuring Real-Time Rating to Find Dropped Calls and Continuation Calls

To configure real-time rating to find dropped calls and continuation calls, perform these tasks:

1. Specify the termination causes that qualify as a dropped call. See "Specifying the Termination Causes for Dropped Calls".
2. Specify how BRM identifies continuation calls. See "Specifying the Rules for Finding Continuation Calls".

Specifying the Termination Causes for Dropped Calls

BRM stores a call session's termination cause in the PIN_FLD_TERMINATE_CAUSE field of the **/active_session** object. For each service type that supports dropped calls, you must specify which field values qualify as a dropped call by editing a service-specific **pin_telco_aaa_params.xml** file. You then load the file into the BRM database's **/config/aaa/gsm/***xxx* object by running the **load_pin_telco_aaa_params** utility.

For more information, see "load_pin_telco_aaa_params".

Table 46–2 lists the **pin_telco_aaa_params.xml** file to use for each service type and the configuration object in which the data is stored. You can find these files in the *BRM_Home*/**sys/data/config** directory.

Table 46–2 pin_telco_aaa_params.xml File

Service Type	Service-Specific pin_telco_aaa_ params.xml File	Configuration Object
GSM data	**pin_telco_gsm_data_aaa_params.xml**	**/config/aaa/gsm/data**
GSM fax	**pin_telco_gsm_fax_aaa_params.xml**	**/config/aaa/gsm/fax**
GSM SMS	**pin_telco_gsm_sms_aaa_params.xml**	**/config/aaa/gsm/sms**

Table 46–2 (Cont.) pin_telco_aaa_params.xml File

Service Type	Service-Specific pin_telco_aaa_ params.xml File	Configuration Object
GSM telephony	**pin_telco_gsm_telephony_aaa_ params.xml**	**/config/aaa/gsm/telephony**
Custom service types	**pin_telco_aaa_params.xml**	**/config/aaa**

To specify the termination causes that qualify as a dropped call, perform these tasks for each service type that supports dropped calls:

1. Open the appropriate service-specific **pin_telco_aaa_params.xml** file in a text editor.
2. Edit the **DroppedCallCause** XML entry. This entry specifies the PIN_FLD_ TERMINATE_CAUSE field values that qualify as a dropped call. Add a **DroppedCallCause** entry for each value that qualifies as a dropped call.

```
<TerminationCauseInfo>
   <DroppedCallCause>4</DroppedCallCause>
</TerminationCauseInfo>
<TerminationCauseInfo>
   <DroppedCallCause>5</DroppedCallCause>
</TerminationCauseInfo>
```

> **Note:** The values correspond to the 3GPP circuit switch (32.005) and packet switch (32.015) codes.

3. Save and close the file.
4. Load the file into the BRM database by using the **load_pin_telco_aaa_params** utility:

```
load_pin_telco_aaa_params -f pin_telco_aaa_params_file
```

   where *pin_telco_aaa_params_file* is the name and location of the service-specific **pin_telco_aaa_params.xml** file. The file name must include the **.xml** extension.

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

To verify that the data loaded correctly, display the service-specific **/config/aaa/gsm/***xxx* 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*.

Specifying the Rules for Finding Continuation Calls

When BRM authorizes or ends a prepaid call, the BRM AAA opcodes determine if the current call is a continuation call by calling the PCM_OP_TCF_AAA_DETECT_ CONTINUATION_CALL helper opcode at the TAG_SESSION processing stage.

The PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL helper opcode calls, at the MATCH_CONTINUOUS_CALL processing stage, the policy opcode(s) specified in the **/config/opcodemap/tcf** object. By default, **/config/opcodemap/tcf** is configured to call the PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL policy opcode. The policy opcode contains the default logic and criteria for finding continuation calls.

- To control which opcodes call the helper opcode and the service types that are supported, modify the **pin_config_opcodemap_tcf** file. See "Configuring BRM to Call the Helper Opcode".
- To change the criteria for finding continuation calls, create a custom policy opcode and configure PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL to call your custom policy opcode at the MATCH_CONTINUOUS_CALL processing stage. See "Specifying the rules for finding continuation calls".

Configuring BRM to Call the Helper Opcode

By default, the following opcodes call the PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL helper opcode at the TAG_SESSION processing stage when processing GSM telephony events:

- PCM_OP_TCF_AAA_AUTHORIZE
- PCM_OP_TCF_AAA_STOP_ACCOUNTING

You can modify which opcodes call the helper opcode or support additional service types by editing the **pin_config_opcodemap_tcf** configuration file. However, if you do modify which opcodes call the helper opcode, you must configure both the opcode that creates the **/active_session** object and the stop accounting opcode to call PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL at the TAG_SESSION processing stage. Although the **/active_session** object can be created by any Services Framework AAA opcode, the object is created only once and is reused by the other Services Framework AAA opcodes. BRM identifies dropped calls when the **/active_session** object is created and deletes redundant **/active_session** objects when the session ends.

Important: If you configure only PCM_OP_TCF_AAA_STOP_ACCOUNTING to call the helper opcode, you must set the **/config/aaa/gsm/***xxx* object's **DeletedFlag** to **True**.

To modify the opcodes or service types, perform these tasks:

1. Open the *BRM_Home***/sys/data/config/pin_config_opcodemap_tcf** configuration file in a text editor.
2. (Optional) Configure an opcode to call a helper opcode by editing the **Framework_Opcode** line. For example, to configure PCM_OP_TCF_AAA_REAUTHORIZE to call a helper opcode, set **Framework_Opcode** to **PCM_OP_TCF_AAA_REAUTHORIZE**.

   ```
   #Framework_Opcode: PCM_OP_TCF_AAA_REAUTHORIZE
   #Processing_Stage: TAG_SESSION
   #Opcode_Map:/service/telco/gsm/telephony, PCM_OP_TCF_AAA_DETECT_CONTINUATION_
   CALL
   ```

3. (Optional) Add an **Opcode_Map** line for each service type that supports the dropped calls feature. For example, to configure PCM_OP_TCF_AAA_STOP_ACCOUNTING to call PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL when processing GPRS services, add the following line:

   ```
   Framework_Opcode: PCM_OP_TCF_AAA_STOP_ACCOUNTING
   Processing_Stage: TAG_SESSION
   Opcode_Map:/service/telco/gsm/telephony, PCM_OP_TCF_AAA_DETECT_CONTINUATION_
   CALL
   Opcode_Map:/service/telco/gprs, PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL
   ```

4. Save and close the file.

5. Load the file into the BRM database by using the **load_aaa_config_opcodemap_tcf** utility:

```
load_aaa_config_opcodemap_tcf -i|-f pin_config_opcodemap_tcf
```

> **Note:** To replace the entire contents of the **/config/opcodemap/tcf** object, use the **-f** parameter. To append data to the object, use the **-i** parameter.

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

To verify that the opcode mappings were loaded correctly, display the **/config/opcodemap/tcf** 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*.

Customizing the Criteria for Finding Continuation Calls

To change the criteria for finding continuation calls, you must create a custom policy opcode. The policy opcode can examine specific flist fields, determine whether an event meets the criteria for a continuation call, and then return to the helper opcode the PIN_FLD_RESULT output flist field set to the following:

- **0** to indicate that the current call *is not* a continuation call.
- **1** to indicate that the current call *is* a continuation call.
- **2** to indicate that the current call *is not* a continuation call because the maximum time duration or maximum number of intermediate calls between a dropped call and a continuation call was exceeded.

For information on creating custom policy opcodes, see "Writing a Custom Facilities Module" in *BRM Developer's Guide*.

- You must then configure BRM to call your custom policy opcode, either *after* or *instead of* the PCM_OP_TCF_AAA_POL_MATCH_CONTINUATION_CALL policy opcode.

To configure the helper opcode to call your custom policy opcode at the MATCH_CONTINUOUS_CALL processing stage, perform these tasks:

1. Open the *BRM_Home***/sys/data/config/pin_config_opcodemap_tcf** configuration file in a text editor.
2. Change the **Opcode_Map** line to reference your custom opcode. For example, to call PCM_OP_YOUR_CUSTOM_OPCODE, enter the following:

```
Framework_Opcode: PCM_OP_TCF_AAA_DETECT_CONTINUATION_CALL
Processing_Stage: MATCH_CONTINUOUS_CALL
Opcode_Map:/service/telco, PCM_OP_YOUR_CUSTOM_OPCODE
Opcode_Map:/service/telco/gsm, PCM_OP_YOUR_CUSTOM_OPCODE
```

> **Note:** Make sure you add an **Opcode_Map** line for each service type that supports the dropped calls feature.

3. Save and close the file.
4. Load the file into the BRM database by using the **load_aaa_config_opcodemap_tcf** utility:

```
load_aaa_config_opcodemap_tcf -i|-f pin_config_opcodemap_tcf
```

Note: To replace the entire contents of the **/config/opcodemap/tcf** object, use the **-f** parameter. To append data to the object, use the **-i** parameter.

5. Stop and restart the CM. See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide*.

To verify that the opcode mappings were loaded correctly, display the **/config/opcodemap/tcf** 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*.

Purging Old Call Data from Memory

To purge old **/active_session** objects from IMDB Cache or the BRM database, use the following command:

```
pin_clean_asos -object "object_type" [-expiration_time number_of_hours] [-state state_value]
```

For more information, see "pin_clean_asos" in *BRM System Administrator's Guide*.

To purge old call data from pipeline memory and the dropped calls data file, use the **RemoveLimit** semaphore file entry:

```
ifw.Pipelines.ALL_RATE.Functions.FunctionPool.DroppedCall.RemoveLimit = 7
```

For more information, see "FCT_DroppedCall" in *BRM Configuring Pipeline Rating and Discounting*.